

**GRAPH APPROXIMATION:
ISSUES AND COMPLEXITY**

A THESIS
Presented to
The Academic Faculty

by

Steven Bradish Horton

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Operations Research

Georgia Institute of Technology
May 1991

**GRAPH APPROXIMATION:
ISSUES AND COMPLEXITY**

APPROVED:

R. G. Parker, Chairman

J. H. Vande Vate

D. C. Llewellyn

Date Approved by Chairperson_____

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENT	iii
LIST OF TABLES	iv
LIST OF FIGURES	v
Chapter	
I. INTRODUCTION	1
A. Graph Approximation	
B. Outline of Thesis	
II. RECURSIVE GRAPHS: ALGORITHMS AND APPROXIMATIONS	4
A. Background	
B. Spanning Eulerian Subgraphs on Halin Graphs	
C. Approximation Issues	
III. COMPLEXITY OF SUBGRAPH AND SUPERGRAPH CONSTRUCTIONS	34
A. Background	
B. Subgraph Results	
1. Series-Parallel	
2. Halin	
C. Supergraph Results	
IV. CONCLUSIONS	59
APPENDIX	61
REFERENCES	81

ACKNOWLEDGEMENT

I would like to thank the many people who were instrumental to my successfully completing this thesis.

First and foremost must be my thesis advisor, Dr R. Gary Parker. He maintained genuine interest in my work throughout the course of research. He was always available to keep my effort focused in the right direction.

I also would like to thank Dr Vande Vate and Dr Llewellyn for their helpful comments and for their participation on my reading committee.

Without the assistance of CPT Stan Haines, the figures in this thesis would not have been as clear and explanatory.

Finally, I thank my wife Julie for her patience and understanding throughout this sometimes difficult period.

This thesis is dedicated to my parents.

LIST OF TABLES

Table	Page
II.B.1. Composition Rules for Operation Type 2	18
II.B.2. Solution Summary	22
APP.A.1. Composition Rules for Operation Type 1	62
APP.A.2. Composition Rules for Operation Type 2	63
APP.A.3. Composition Rules for Operation Type 3	63
APP.A.4. Composition Rules for Operation Type 4	64
APP.A.5. Composition Rules for Operation Type 5	64
APP.A.6. Composition Rules for Operation Type 6	65
APP.A.7. Composition Rules for Operation Type 7	65
APP.B.1. Composition Rules for Operation Type 1	72
APP.B.2. Composition Rules for Operation Type 2	73
APP.B.3. Composition Rules for Operation Type 3	73
APP.B.4. Composition Rules for Operation Type 4	74
APP.B.5. Composition Rules for Operation Type 5	74
APP.B.6. Composition Rules for Operation Type 6	75
APP.B.7. Composition Rules for Operation Type 7	75
APP.B.8. Solution Summary	76
APP.B.9. Solution Summary	77

LIST OF FIGURES

Figure	Page
II.B.1.	8
II.B.2.	8
II.B.3.	11, 12
II.B.4.	15
II.B.5.	15
II.B.6.	15
II.B.7.	16
II.B.8.	16
II.B.9.	19
II.B.10.	24
II.C.1a.	27
II.C.1b.	27
II.C.1c.	27
II.C.2a.	29
II.C.2b.	29
II.C.3.	31
II.C.4.	31
II.C.5.	32
III.B.1a.	39
III.B.1b.	39
III.B.1c.	41

III.B.1d.	41
III.B.2a.	46
III.B.2b.	46
III.B.2c.	50
III.B.2d.	50
III.C.1.	54
III.C.2.	56
APP.A.1.	66
APP.A.2.	66
APP.A.3.	67
APP.A.4.	67
APP.A.5.	68
APP.A.6.	68
APP.A.7.	69
APP.B.1.	78-80

CHAPTER I

INTRODUCTION

A. Graph Approximation

It is well known that many problems are formally intractable on arbitrary graphs. It is also often the case that these problems can be solved by polynomial time algorithms when instances are confined to special classes. For example, the subclass of planar graphs might yield to a fast algorithm for a problem which is otherwise difficult. Bipartite graphs are also known to act as suitable special cases as do trees, claw-free graphs, and others.

In this regard, much recent attention has been given to the class of so-called recursively constructed graphs [4, 5]. Among these are series-parallel graphs, Halin graphs, and partial k -trees. Important here is that an enormous number of problems are solvable (often in linear time) on such recursive graph classes by employing (understandably) dynamic programming-like strategies. In fact, these strategies are so well understood that various formal models of the inherent recursive computation have been constructed [4].

It is precisely this success in dealing with recursive graph classes that leads us to a fairly natural question; one which provides motivation for the work reported in this thesis: Given a graph which is not a member of any (known) recursive class, how reasonable is it to attempt to approximate the graph by one which is in a well-solved recursive class? The purpose of this approximation is to solve a given problem on the latter and transform its solution so as to be admissible on the original structure. For example, a graph might be, in some sense, "almost" series-parallel. A maximum/maximal subgraph is formed which is series-parallel, the problem of interest is solved accordingly, and the solution is "patched" back in to the original graph.

Of course, there are some obvious issues that emerge in this notion of graph approximation. Are the approximations dependent or influenced by the problem? Upon what sort of recursive class should we base our approximation? Should approximating subgraphs or supergraphs be sought and in either case, can they be exhibited efficiently? The aim of this research is to examine these sorts of questions.

B. Outline of Thesis

In Chapter II, we describe some fundamental properties and results pertaining to recursive graph classes. We then demonstrate the problem solving strategy on these graphs by

giving a procedure for solving the spanning eulerian subgraph problem on the recursive class of Halin graphs. We conclude with issues that are pertinent in terms of employing the sort of approximations described above.

In Chapter III, we present some negative results. Specifically, we establish the difficulty of producing certain natural approximations for two well-known recursive classes: series-parallel, and Halin graphs. In the case of Halin graphs, we prove that both a subgraph as well as a supergraph approximation is hard to find. The final chapter describes some directions for further work.

CHAPTER II

RECURSIVE GRAPHS: ALGORITHMS AND APPROXIMATIONS

A. Background

Informally, a recursive graph class is one in which any sufficiently large member can be composed by joining smaller members in the class at specific vertices called **terminals**. Letting the number of terminals be k , we often refer to these as **k -terminal graphs**. More formally, a k -terminal graph $G = (V, T, E)$ has a vertex set V , an edge set E , and a (possibly ordered) set of distinguished vertices or terminals $T \subseteq V$ such that $T = \{t_1, t_2, \dots, t_{t(G)}\}$, where $t(G) = |T| \leq k$. For some k , let U be the set of k -terminal graphs. Then, a **recursively constructed graph family** $F = (B, R)$ in U has base elements $B \subseteq U$ and a finite set of recursive **composition** operations $R = \{R_1, R_2, \dots, R_n\}$, where each $R_i: U^p \rightarrow U$. Here, p refers to the arity of R_i . Generally, we consider only base elements in which all vertices are terminals. However, it is easy to see that all such structures decompose trivially into edges, so we often take B to be a singleton consisting of K_2 .

The notion of composition can be described by the same general form. For $1 \leq i \leq m$, let $G_i = (V_i, T_i, E_i) \in U$, where V_1, V_2, \dots, V_m are mutually disjoint. Let $G = (V, T, E) \in U$ as well. A valid vertex mapping is a function $f: \cup_{1 \leq i \leq m} V_i \rightarrow V$ such that four conditions are satisfied. First, vertices from the same G_i must remain distinct after composition. Second, only terminals can map to terminals. Third, only terminal vertices can merge, and last, edges are preserved upon composition. If f is a valid vertex mapping, then we shall write the corresponding m -ary composition operation as $f(G_1, G_2, \dots, G_m) = G$.

A **decomposition tree** of a k -terminal graph G is a rooted tree with vertex labels g and f such that

- $g_v = G$ if v is the root,
- $f_v \in R$ if v is an interior node,
- $g_v = f_v(g_{v_1}, g_{v_2}, \dots, g_{v_m})$ if interior node v has children v_1, \dots, v_m , and
- $g_v \in B$ if v is a leaf.

Decomposition trees are very important in the underlying strategy of problem solving on k -terminal recursive graphs. If we know the solution to a given problem (i.e. vertex cover, dominating set, etc.) on the leaf graphs of a decomposition tree (base graphs), then the postorder traversal of the tree with appropriate recurrence formulae

(relevant to the given problem) would produce an efficient algorithm for the problem on the given k -terminal graph.

To develop appropriate recurrence relations for a dynamic programming solution, one starts by building a multiplication table f' for each composition operation f . If $G = f(G_1, G_2)$ then the multiplication table f' exhibits the outcome for G that corresponds to each pair of compatible subgraph property vectors for G_1 and G_2 . It is now straightforward to construct the recurrence relations directly from the multiplication tables. These formulae simply compute the optimal property values from among the compositions of the compatible pairs [3,4].

B. Spanning Eulerian Subgraphs on Halin Graphs

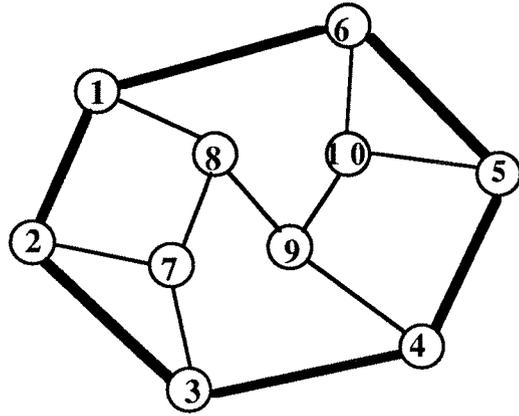
We can demonstrate the basic algorithmic strategy on k -terminal graphs by considering the following problem: Given a graph $G=(V,E)$ and an integer K , does G possess a spanning eulerian subgraph of size (edge cardinality) at least K ? Referring to this problem as P_{SES} , suppose our interest is in solving it on the particular recursive class of Halin graphs. This is meaningful, since P_{SES} is known to be hard in general (in fact it is hard on cubic, planar graphs [10]). First we consider the structure of Halin graphs.

Halin graphs are planar with the property that their edge sets can be partitioned into a spanning tree, L , having no degree-2 vertices, and a cycle, C , on the leaves of L .

We shall call these leaves **pendants**. A Halin graph appears in Figure II.B.1 with L and C denoted. These graphs have some interesting properties which we return to in the next chapter. Our interest at this point, however, is confined to their recognition as members of a well-solved class. In particular, Borie et al [3] demonstrated that Halin graphs are contained in a three terminal class and can, moreover, always be decomposed (efficiently) by the repeated application of a finite set of binary operations. Therefore, if G is a Halin graph we may, without loss of generality, assume its decomposition tree to be part of any associated problem instance.

Let G be a Halin graph (observe that we can do so without loss of generality since testing for the Halin property is easy [5]), and further let us assume a plane embedding of G as shown in Figure II.B.2. Here vertex t_1 is a terminal corresponding to any nonleaf vertex of L . The other terminals are given by t_2 and t_3 and correspond to the rightmost and leftmost leaves of L as indicated.

Our decomposition of G follows by applying a generalization of the well-known series and parallel operations. We leave specific details of these in [3]; however recall that in a series operation, certain vertices lose (gain) terminal status after composition



Cycle edges in
bold face

Figure II.B.1

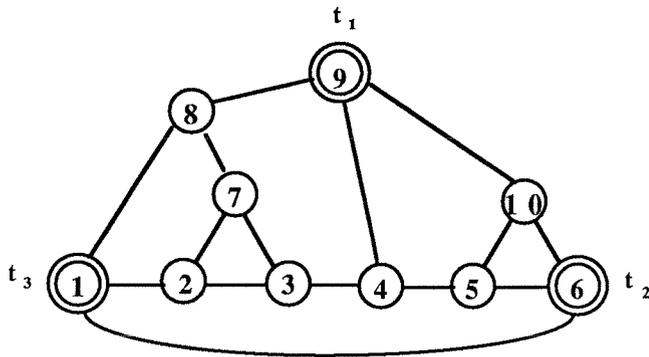


Figure II.B.2

(decomposition) while for a parallel operation, such status is preserved.

Now, let us define a **hypoHalin graph** to be a planar graph whose edge set can be partitioned into a spanning tree and a path from its leftmost to its rightmost leaf (we assume a plane embedding exists as described above) and which passes through the other leaves of the tree. Then if G is a Halin graph with terminals t_1 , t_2 , and t_3 (with the stated embedding), G can be decomposed by a parallel operation into a K_2 with edge $e = \{t_2, t_3\}$ and a hypoHalin graph $G' = G \setminus e$. Our claim is that any nontrivial hypoHalin graph is decomposable into two smaller hypoHalin graphs or spanning subgraphs of hypoHalin graphs by either the generalized series or parallel operations.

For a given hypoHalin graph G' or a spanning subgraph thereof, we can eliminate any edge with both of its vertices terminal by a parallel operation. Otherwise, let P be a leaf (not t_2 or t_3) which is nearest to t_2 such that it is reachable from t_3 by a path that passes through neither other leaves nor t_1 . Then decompose G' by a series operation into two graphs, one with terminals t_1 , t_2 , and P and the other having terminals t_1 , P , and t_3 . Such a decomposition must be possible since there is only one path from P to t_1 which does not include leaf vertices (other than P itself).

If the stated vertex P does not exist, then G' must possess only terminal vertices and is therefore trivially decomposable into edges by successive parallel operations or else possesses degree-1 vertices, which can be eliminated using series operations. We must also be specific when operations are applied to fewer than three terminals. Regardless, the decomposition proceeds in this manner, stopping when all leaves are single edges. A decomposition tree for the Halin graph in Figure II.B.1 is given in Figure II.B.3.

Now, let us return to P_{SES} . In order to develop an algorithm suitable for any Halin graph, we need to construct the multiplication tables corresponding to the series and parallel composition operations. To this end, we must determine suitable properties relative to the terminals of the component subgraphs that allow us to decide whether or not a spanning eulerian subgraph exists for a given instance, and if so, to ascertain exactly which edges make up this subgraph. It is an obvious exercise to deal with cardinalities. Consider the following properties:

- 1) A spanning eulerian subgraph of G_i . We will use the notation $[SES]$ to indicate this property.

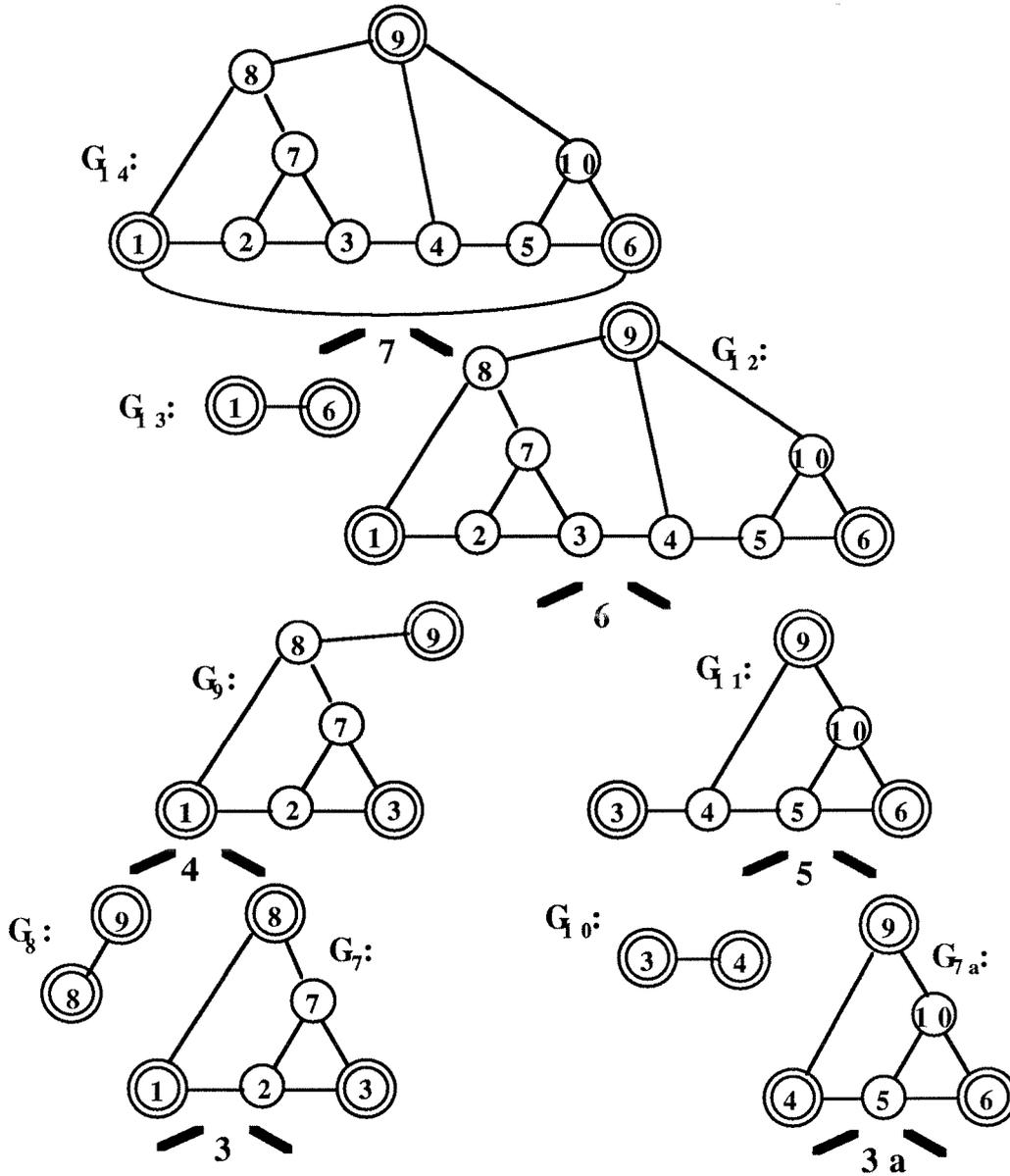


Figure II.B.3

Continued

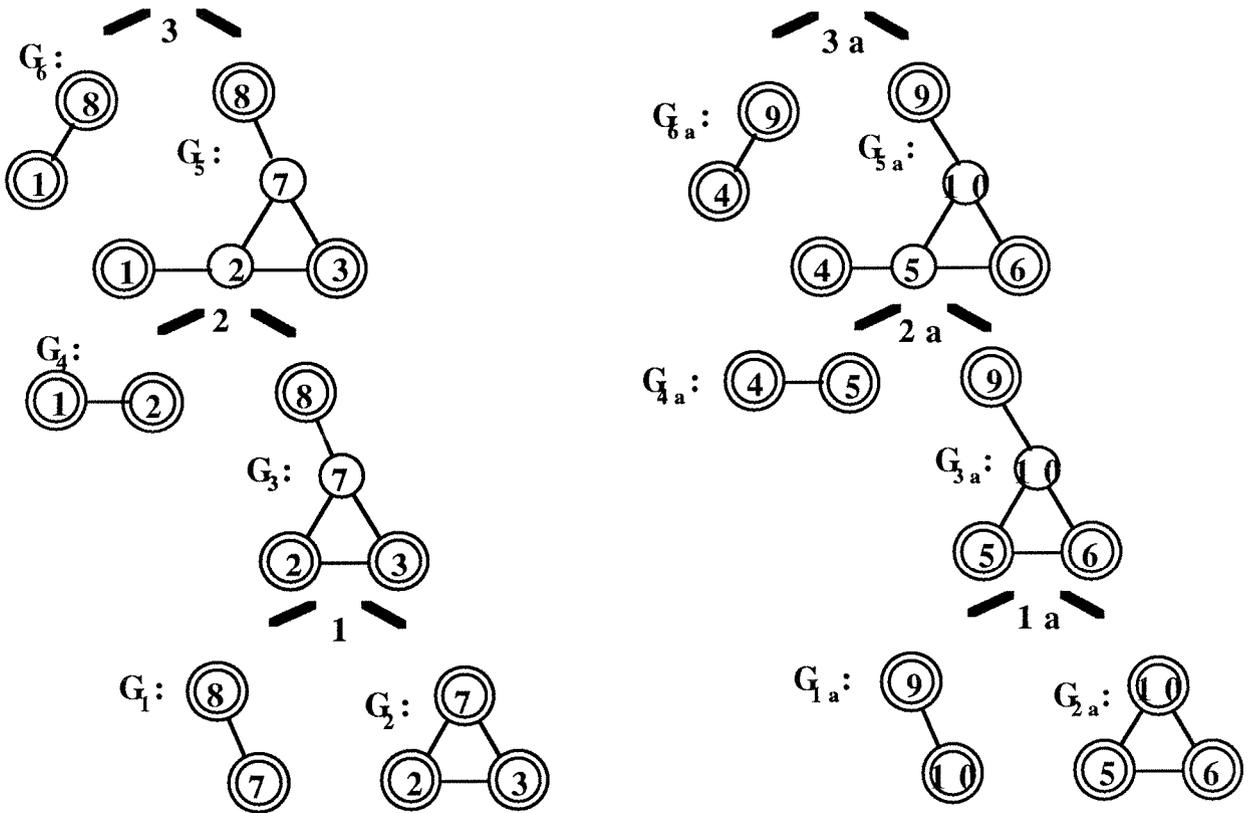


Figure II.B.3

2) A spanning subgraph of G_i consisting of three (possibly nonsimple) cycles; one incident to each terminal, but none incident to more than one. We use the notation [TRI] to describe this property.

3) A spanning subgraph of G_i consisting of two (possibly nonsimple) cycles; one incident to exactly two terminals and one incident to only the other terminal. This can happen three different ways, since the cycle on two terminals can extend between the left and top terminals, the right and top terminals, or the left and right terminals. We use the notation [LTBI], [RTBI], and [LRBI] to describe these properties.

4) A spanning subgraph of G_i consisting of a (possibly nonsimple) path connecting two terminals and a (possibly nonsimple) cycle incident to only the third terminal. As before, there are three such possibilities. We shall call these properties "paths" and use the notation [LTPA], [RTPA], and [LRPA].

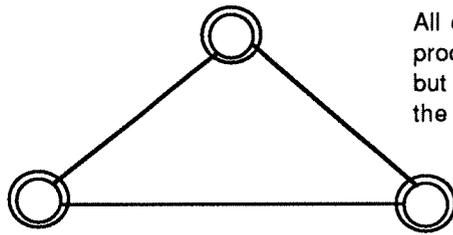
5) A spanning subgraph of G_i consisting of a (possibly nonsimple) path starting at one terminal, passing through another terminal, and finishing at the third terminal. Again there are three such

possibilities which we call "forks" and denote by [LFRK], [TFRK], and [RFRK], where the specified terminal is the terminal that is of even degree in the (path) subgraph. Note that since this path can be nonsimple, it can pass **through** the end-of-path terminals in addition to ending there. Thus an admissible [LFRK] might start at the top terminal, go from there to the left terminal, then back to the top terminal, and finally end at the right terminal. (This is called a "fork" because it can be thought of as two edge disjoint paths starting at the named terminal, each ending at a different terminal, and thereby connecting all three terminals).

Illustrations of these subgraph properties are given in Figures II.B.4 - II.B.8.

Although Halin graphs are efficiently decomposable into base graphs which are edges, we will, as a convenience, halt our decomposition when either a K_2 or a K_3 is reached. As a consequence, we establish initialization rules for both of these cases.

A single edge can be either **in** a subgraph or **out** of a subgraph, so we assign a value to these two conditions. Since we seek a maximum cardinality spanning eulerian



All edges in these Figures may proceed through other vertices, but they must begin and end at the Terminal vertices as shown.

Figure II.B.4
Spanning Eulerian Subgraph
(SES)

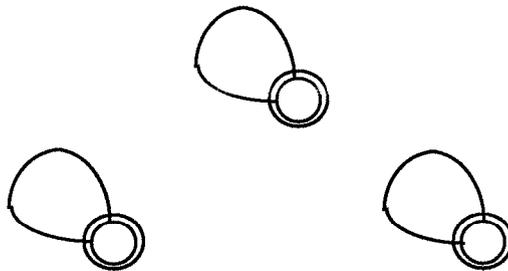


Figure II.B.5
(TRI)

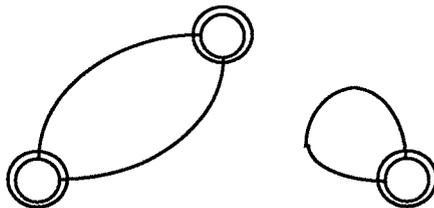


Figure II.B.6
(BI)
(LTBI in this case)

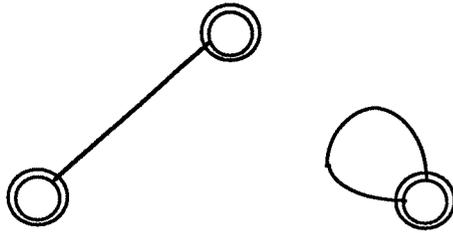


Figure II.B.7
(Path)
(LTPA in this
case)

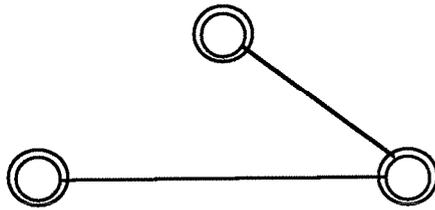


Figure II.B.8
(Fork)
(RFRK in this
case)

subgraph, each edge in the subgraph contributes 1 towards this goal, while an edge out of the subgraph contributes 0.

Thus we have

- $P_{IN}[K_2] = 1$ and
- $P_{OUT}[K_2] = 0$.

The case of the K_3 is only slightly more complicated. The K_3 can take on most of the properties of our general three terminal cases. Following, we examine each of the possible properties:

- 1) $P_{SES}[K_3] = 3$, with all edges included in the subgraph.
- 2) $P_{TRI}[K_3] = 0$, with no edges included.
- 3) $P_{LTBI}[K_3] = P_{RTBI}[K_3] = P_{LRBI}[K_3] = -\infty$, since these conditions cannot be met with a K_3 .
- 4) $P_{LTPA}[K_3] = P_{RTPA}[K_3] = P_{LRPA}[K_3] = 1$, including only one edge in each case.
- 5) $P_{LFRK}[K_3] = P_{TFRK}[K_3] = P_{RFRK}[K_3] = 2$, including two edges in each case.

Note that by simply substituting edge weights for the above values, the maximum weighted spanning eulerian subgraph problem can be easily solved as well.

Now, since these series and parallel operations take place on certain subsets of the terminal set, we will form seven multiplication tables. We will present one such table

here and demonstrate how it is derived. The complete set of seven tables appears the appendix.

Consider the series operation where K_2 is merged with the vertex t_1 of a three terminal graph (Operation Type 2). Note that the operation must be specifically defined; here the right vertex of the K_2 (G_1) merges with the top vertex of G_2 . Also observe that this vertex then loses its terminal status in the resultant composed graph (see Figure II.B.9). The following multiplication table represents this operation:

Table 2.B.1 Composition Rules for Operation Type 2

$G_2 \rightarrow$ $G_1 \downarrow$	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK
IN	-	-	-	-	-	LTPA	RTPA	-	LFRK	-	RFRK
OUT	LRBI	-	TRI	TRI	-	-	-	-	-	LRPA	-

This table is derived simply by attempting to combine G_1 and G_2 in the manner described above, with each possible property as indicated in the table. For example, if the edge of G_1 was **in** a subgraph, and the edges in the subgraph of G_2 formed a **TRI**, we would observe the cell in row 1,

column 2 of the table II.B.1. To represent these properties, we would write $P_{IN}[G_1]$ and $P_{TRI}[G_2]$.

Now consider the case of $P_{OUT}[G_1]$ and $P_{LTBI}[G_2]$. This produces a TRI for the resultant composed graph since the connection between the left and top terminals that existed in G_2 is severed by the addition of the **out** edge from G_1 (this case is possible since the even degree requirement is maintained at the newly "buried" terminal). Observe that a "-" entry in the table signifies incompatible subgraph pairs.

Once the multiplication tables are constructed, recurrence formulae need to be derived for each table. The following formulae apply to the above table:

$$\begin{aligned}
 P_{SES}[G] &= -\infty \\
 P_{TRI}[G] &= \text{Max}\{P_{OUT}[G_1] + P_{LTBI}[G_2], P_{LTBI}[G_1] + P_{RTBI}[G_2]\} \\
 P_{LTBI}[G] &= -\infty \\
 P_{RTBI}[G] &= -\infty \\
 P_{LRBI}[G] &= P_{OUT}[G_1] + P_{SES}[G_2] \\
 P_{LTPA}[G] &= P_{IN}[G_1] + P_{LTPA}[G_2] \\
 P_{RTPA}[G] &= P_{IN}[G_1] + P_{RTPA}[G_2] \\
 P_{LRPA}[G] &= P_{OUT}[G_1] + P_{TFRK}[G_2] \\
 P_{LFRK}[G] &= P_{IN}[G_1] + P_{LFRK}[G_2] \\
 P_{TFRK}[G] &= -\infty \\
 P_{RFRK}[G] &= P_{IN}[G_1] + P_{RFRK}[G_2]
 \end{aligned}$$

This computation is made explicit by solving P_{SES} on the Halin graph shown in Figure II.B.1. We summarize the work in table II.B.2. Note that:

- $P[G] = (In, Out)$ for the case of K_2 , and
- $P[G] = (SES, TRI, LTBI, RTBI, LRBI, LTPA, RTPA, LRPA, LFRK, TFRK, RFRK)$ for the every other case.

Note also that connections 1, 2, and 3 are identical to connections 1a, 2a, and 3a, respectively, except for the designations of the edges involved. Entries that are underlined will be seen to aid in solution retrieval.

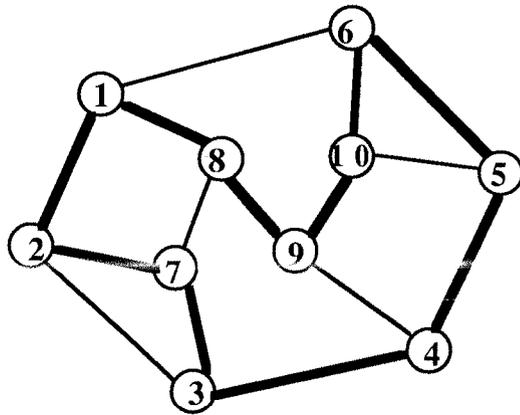
Table II.B.2 Solution Summary

Graph	Graph Properties
P[G ₁]	(1, <u>0</u>)
P[G _{1a}]	(<u>1</u> , 0)
P[G ₂]	(3, 0, -∞, -∞, -∞, 1, 1, 1, 2, <u>2</u> , 2)
P[G _{2a}]	(3, 0, -∞, -∞, -∞, 1, 1, 1, 2, 2, <u>2</u>)
P[G ₃]	(-∞, -∞, -∞, -∞, 3, 2, 2, <u>2</u> , 3, -∞, 3)
P[G _{3a}]	(-∞, -∞, -∞, -∞, 3, 2, 2, 2, 3, -∞, <u>3</u>)
P[G ₄]	(<u>1</u> , 0)
P[G _{4a}]	(<u>1</u> , 0)
P[G ₅]	(-∞, 3, -∞, -∞, -∞, 3, 3, <u>3</u> , -∞, -∞, 4)
P[G _{5a}]	(-∞, 3, -∞, -∞, -∞, 3, 3, 3, -∞, -∞, <u>4</u>)
P[G ₆]	(<u>1</u> , 0)
P[G _{6a}]	(1, <u>0</u>)
P[G ₇]	(5, 3, 4, -∞, -∞, 4, 3, 3, <u>4</u> , 4, 4)
P[G _{7a}]	(5, 3, 4, -∞, -∞, 4, 3, 3, 4, 4, <u>4</u>)
P[G ₈]	(<u>1</u> , 0)
P[G ₉]	(-∞, 4, -∞, -∞, 5, 5, 4, 4, <u>5</u> , -∞, 5)
P[G ₁₀]	(<u>1</u> , 0)
P[G ₁₁]	(-∞, 4, -∞, 5, -∞, 5, 4, 4, -∞, 5, <u>5</u>)
P[G ₁₂]	(<u>10</u> , 9, 10, 10, -∞, 9, 9, 8, 10, 9, 10)
P[G ₁₃]	(1, <u>0</u>)
P[G ₁₄]	(<u>10</u> , 9, 10, 10, 9, 9, 9, 10, 11, 11, 11)

From the computation of $P[G_{14}]$, we see that the largest spanning eulerian subgraph of the given instance has 10 edges (and that the largest **tri** has 9 edges, etc). We simply retrace our steps to determine which edges from the basic K_2 's and K_3 's are **in** the spanning eulerian subgraph and which ones are **out**.

As stated above, the underlined entries demonstrate the "backtracking" method of retrieving the solution. For example, $P_{SES}[G_{14}]$ was formed by $P_{SES}[G_{12}]$ and $P_{OUT}[G_{13}]$. Hence, edge (1,6) is **out** of the spanning eulerian subgraph. Continuing in this fashion will produce the stated subgraph as indicated by the bold edges in Figure II.B.10.

It is clear that this algorithm operates in linear time because there is only a constant amount of information to be computed for each node of the decomposition tree, and the size of this decomposition tree is linear in the cardinality of the edge set of G . It should also be obvious that a similar method will work for any class of recursively constructed k -terminal graphs for fixed k , once the decomposition tree for a graph in the class is found. Note that the value of k must be bounded. Indeed, if this were not so, the amount of information required could grow exponentially, or even faster for some problems.



Spanning Eulerian
Subgraph edges in bold

Figure II.B.10

C. Approximation Issues

Now, we turn our attention to issues that must inevitably be raised in any attempt to generate heuristic solutions for arbitrary graphs by employing an approximation approach as described in sections A and B above. The first of these involves examining the problem of finding a subgraph or supergraph of a given instance graph that is recursively constructable. A second issue pertains to questions related to the transformation of solutions on our recursive sub/supergraph approximations back to the original instances. These questions are not unrelated and as we shall see in the next chapter, certain of them may not be easy to resolve.

The notion of finding an approximating recursive graph of one not so structured can certainly invite uninteresting strategies. For example, assuming the original graph is at least connected, we could always "approximate" it with a spanning tree. Most problems are well-solved on trees, but this would likely be of little consequence when the tree solution was applied to the original graph. Indeed, it would probably not even be admissible (i.e. vertex cover, chromatic index, etc.). Further, even if admissible, its quality would almost certainly be suspect.

But our interest, of course, is in approximations which are "close" to the original structures. That is we want to

approximate graphs which are "nearly series-parallel" or "nearly Halin" by graphs in the respective recursive classes. It would also seem logical, in say the subgraph case, to prefer a subgraph on edge set E_1 to one on edge set E_2 where $|E_1| \geq |E_2|$ (in the supergraph case the opposite relation would be desirable). Unfortunately, this need not be meaningful either as the following apparent anomaly suggests.

Consider the graph G in Figure II.C.1a and suppose we are solving P_{SES} . Further, let us create a series-parallel subgraph of G to serve as an approximation. Recall that series-parallel graphs are exactly those without subgraphs homeomorphic to K_4 . Moreover, it is well known that biconnected series-parallel graphs are members of a 2-terminal recursive class. In any event, let us create a maximal, 2-terminal series-parallel subgraph of G upon which we solve P_{SES} . This can be done in linear time following a result in [10].

Now, the graph G in Figure II.C.1a has edge cardinality $(7k+11)/2$ where $k \equiv 1 \pmod{4}$. A maximal series-parallel subgraph is easy to find and one such possibility is given by G' in Figure II.C.1b. G' has edge cardinality $3k+5$. Observe also that the solution to P_{SES} on G' is a hamiltonian cycle in G' having $2k+1$ edges. On the other hand, G'' in Figure II.C.1c could have been our maximal

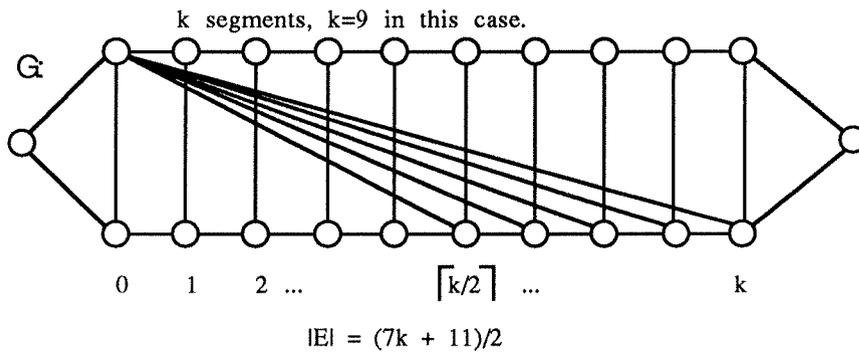


Figure II.C.1a

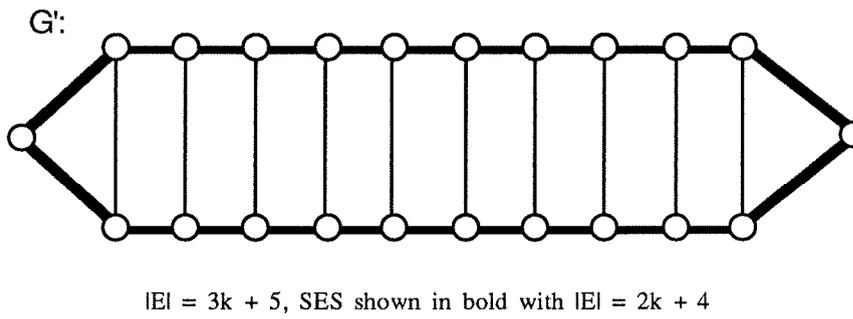


Figure II.C.1b

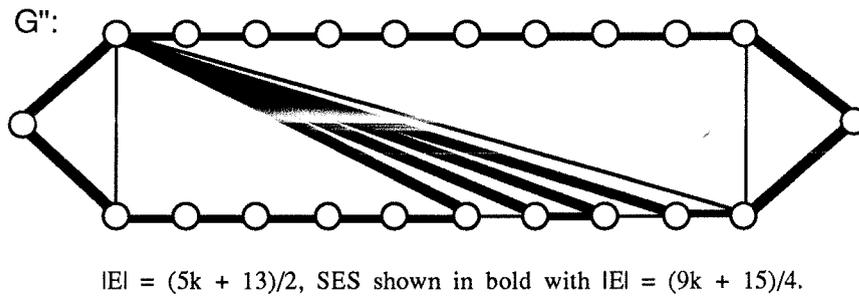


Figure II.C.1c

construction. It has $(5k+13)/2$ edges but, as illustrated, yields a solution to P_{SES} of size $(9k+15)/4$ or on the order of a $k/2$ improvement over G' ! Thus we have a phenomenon where a conceivably better (in one sense) approximation leads to a poorer solution. Clearly, structure can be more important than size.

But there are troublesome issues beyond this. Observe that in our illustration above, we produced a maximal subgraph. That we did so is due to the ease with which this is accomplished for series-parallel subgraphs. That we did not seek a maximum cardinality subgraph (the anomaly notwithstanding) follows, as we show later, from the intractability of finding same.

At least our solution to P_{SES} on a subgraph of G say G^- would be a feasible solution on G . As suggested earlier, however, for a problem like vertex cover this would not generally be true. But if we had a supergraph approximation of G , G^+ , we would not have such a problem since any cover on G^+ would be admissible on G . It is also true, however, that the solution from G^+ might be very poor for G . Consider the vertex cover problem on G in Figure II.C.2a. First, observe that G is not series-parallel and so trivially no (series-parallel) G^+ exists. However, let us create a Halin supergraph of G to act as G^+ . A suitable construction is shown in Figure II.C.2b. Now, on G^+ an

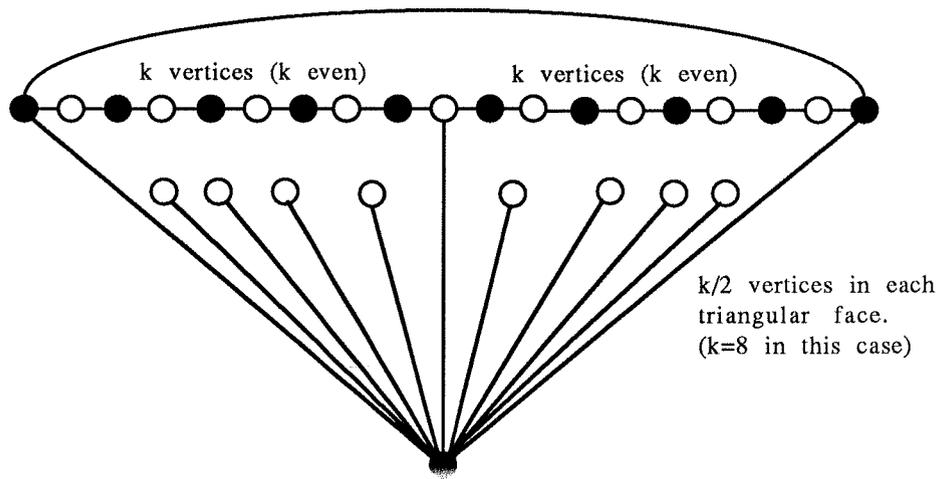


Figure II.C.2a

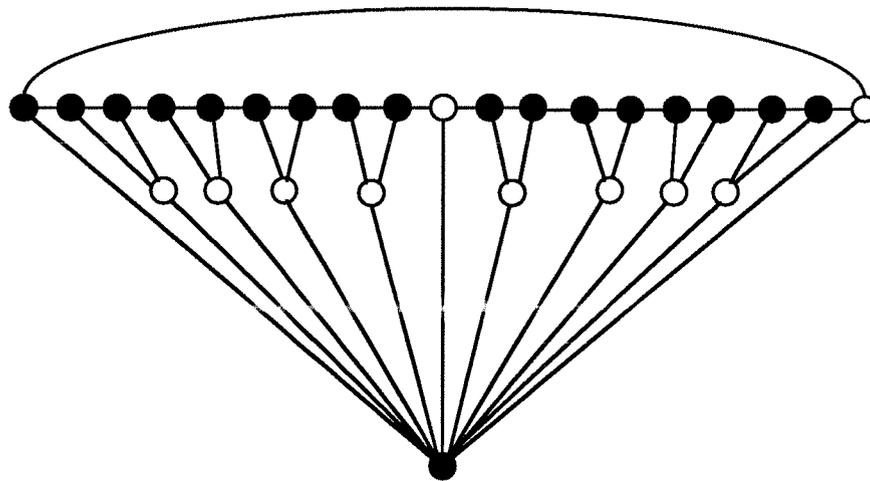


Figure II.C.2b

optimal cover is denoted by the darkened vertices of which there are $2(k+1)$. Unfortunately, this solution is very poor on G which can be covered by at most $k+3$ vertices as shown by the darkened vertices in Figure II.C.2a.

More discouraging than the quality issue, however, is that in the case of at least Halin approximations, forming **any** G^+ can be difficult. Note the distinction on this point with regard to series-parallel graphs. If G is not series-parallel then, as suggested earlier, adding edges cannot rectify this situation. But there are non-Halin graphs which can be made so by adding edges. Conversely, there are also non-Halin graphs where no Halin G^+ can be formed at all.

Consider G in Figure II.C.3. Clearly, G is not Halin (since vertices 5 and 11 are degree 2). But adding edges $(2,5)$, $(2,8)$, and $(11,14)$ creates G^+ which is Halin. This G^+ and its corresponding sets L and C are shown in Figure II.C.4. Solving vertex cover on G^+ produces the vertex set $V_c = \{2,4,5,6,8,9,10,12,14,16\}$ (note that the explicit algorithm for vertex cover on Halin graphs and a solution summary for the above result is given in the appendix). Now suppose we alter the instance by adding edge $(5,11)$ as shown in Figure II.C.5. It is easy to verify that no Halin supergraph is even possible in this case!

Instance Graph $G=(V,E)$

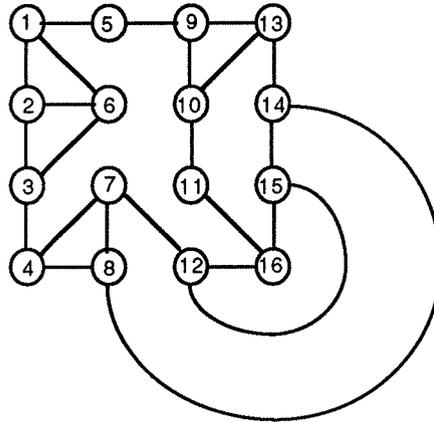


Figure II.C.3

Supergraph G_+

-Added edges in bold: — 2-5, 2-8, 11-14

-Halin cycle more bold: — 1-5-9-10-11-16-12-7-4-3-6-1
(Halin cycle denotes C)

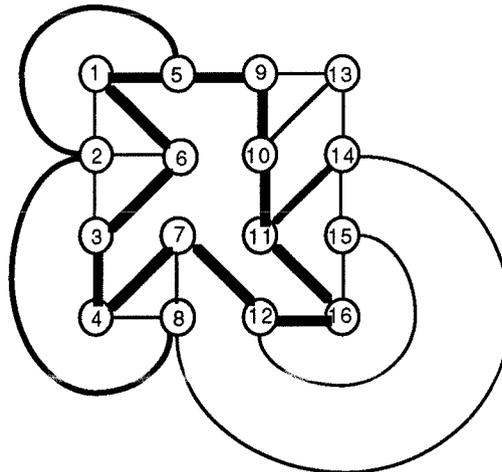


Figure II.C.4

In the next chapter, we provide results which have to be considered discouraging insofar as certain subgraph and supergraph approximations are concerned. Indeed, these results, in some sense, anticipate the sorts of concerns raised and demonstrated in this section.

CHAPTER III

COMPLEXITY OF SUBGRAPH AND SUPERGRAPH CONSTRUCTIONS

A. Background

Series-parallel and Halin graphs were defined previously. Not mentioned before, however, are some properties possessed by Halin graphs which will help us in developing the results of this chapter. Below, we state some of these.

First, Halin graphs are minimally 3-edge connected [5] and as an obvious consequence of this we have:

Property: Subgraphs of Halin graphs are not Halin.



Thus, the class of Halin graphs is not closed under the subgraph operation. That is, given a decomposition tree of a Halin graph, every descendant graph is not Halin. This, of course, is typically not the case with graphs constructed in a recursive fashion. For example, Halin graphs are distinctly different from series-parallel graphs in this regard.

In fact, distinctions between Halin and series-parallel graphs are quite fundamental. We have

Property: No Halin graph is series-parallel.

Proof: The property is immediate following a result of Dirac [4] which asserts that any (simple) graph with minimum vertex degree 3 possesses a subgraph homeomorphic to K_4 and is thus not series-parallel. But a necessary condition for a graph to be Halin is that every vertex degree be at least 3 and the property follows.

■

The number of edges in any Halin graph is easy to determine. If L and C are the tree and cycle edges, respectively, then for a Halin graph of order p and for $|C|=k$ the corresponding total edge cardinality is $p+k-1$. Now, since C passes through only and all the pendants of L , there are exactly k pendants in L . The maximum number of such pendants is $p-1$ which is achieved by star graphs, $K_{1,p-1}$. Hence, the largest Halin graph on p vertices has $2p-2$ edges. Interestingly, this is essentially the same as for series-parallel graphs (edge cardinality of any series-parallel graph is bounded from above by $2p-3$ [9]). The following property comes in handy later:

Property: Let G be Halin with $|C|=k$. Then k is bounded as

$$p/2 + 1 \leq k \leq p - 1$$

Proof: The upper bound was just established. Let k be the size of the smallest cycle. Since G is Halin, there are $p-k$ vertices in L which are not pendants and which are connected by $p-k-1$ edges. Total degree generated by these "tree" edges is therefore $2(p-k-1)$. Also, there are k tree edges that connect pendants to the tree. These add k to the total degree of the non- pendant vertex structure just described for a total of $2p-k-2$. But since every vertex in a Halin graph has degree at least 3, there must exist degree at least $3(p-k)$ contributed by the stated $p-k$ vertices. Thus $3(p-k) \leq 2p-k-2$ which implies $k \geq p/2 + 1$ as claimed.

■

As a consequence, we have that for fixed k , every maximal Halin graph on p vertices has the same size. From the subgraph property given earlier, it is also the case that every such graph is minimal. We now proceed with the key results of this chapter.

B. Subgraph Results

1. Series-Parallel

We direct our attention first to the problem of finding a maximum cardinality biconnected series-parallel subgraph of a graph G . We state the problem in the following way:

P_{SPSUB} : Given a graph $G=(V,E)$ and an integer k , does there exist a biconnected series-parallel subgraph of G having edge cardinality no less than k ?

Theorem: P_{SPSUB} is ~~NP~~-complete.

Proof: We will employ a reduction from the problem of testing which cubic, planar graphs are Hamiltonian [8]. Accordingly, let $G=(V,E)$ be a cubic, planar instance and construct G' as follows: replace each vertex in G with a plane representation of K_4 . For each K_4 "module" so constructed, connect the three vertices incident to the infinite face (of K_4) to the corresponding adjacent module preserving the adjacencies in G . The construction is demonstrated in Figure III.B.1a. Let this graph be $G'=(V',E')$ and set $k = 6|V|$.

Now, assume G is Hamiltonian with cycle $E_c \subseteq E$, and let us form disjoint subsets of E' as A' and B' where A' is the set of edges in E' between the K_4 modules corresponding to E_c and B' is the set of edges inside the modules. We can now form the desired subgraph of G' , say $G^*=(V^*,E^*)$ as follows: set $V^* = V$ and construct E^* as $A' \cup B'^*$ where B'^* contain all edges in B' except one, say $e = \{i,j\}$, from each K_4 module where i and j are not both incident to edges in A' . Each K_4 module loses one edge so $|B'^*| = 5|V|$. Since A' corresponds exactly to E_c we know that $|A'| = |V|$ and so

$|E^*| = 6|V|$. The constructed graph G^* is easily seen to be biconnected and series-parallel. Figure III.B.1b demonstrates.

For the converse, suppose there exists a biconnected series-parallel subgraph of G' , $G^=(V^,E^)$ with $|E^| = 6|V|$. Accordingly, let us partition $E^$ into disjoint sets $A^$ and $B^$ such that $A^$ is the set of "intermodular" edges in set $E^$ and $B^$ is the set of "intramodular" edges in $E^$. $A^$ and $B^$ are nonempty since $G^$ is biconnected and are proper subsets of A' and B' since $G^$ is series-parallel.

First, note that if more than $5|V|$ intramodular edges are in $E^$, then at least one module would exist as K_4 and we deny that $G^$ is series-parallel. Therefore $|B^| \leq 5|V|$.

Now we consider the intermodular edges. Since G is cubic, each module in $G^$ must be incident to 0, 1, 2, or 3 edges in $A^$. Clearly, a module cannot be incident to zero or one edges from $A^$, since otherwise $G^$ would not be biconnected.

Suppose a module is incident to three edges in $A^$. This case will be divided into several sub-cases based on the number of the $B^$ edges in the module. Note that there could not be six edges in a module since G' is series-parallel. Thus, the case for five edges will be handled below, while the case for four or fewer will be addressed

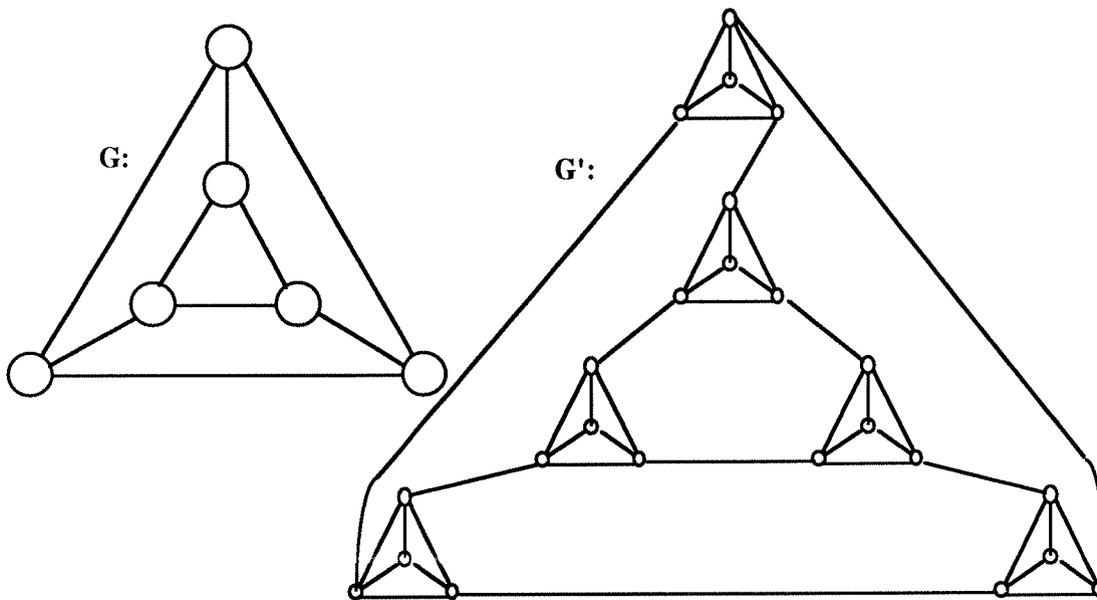


Figure III.B.1a

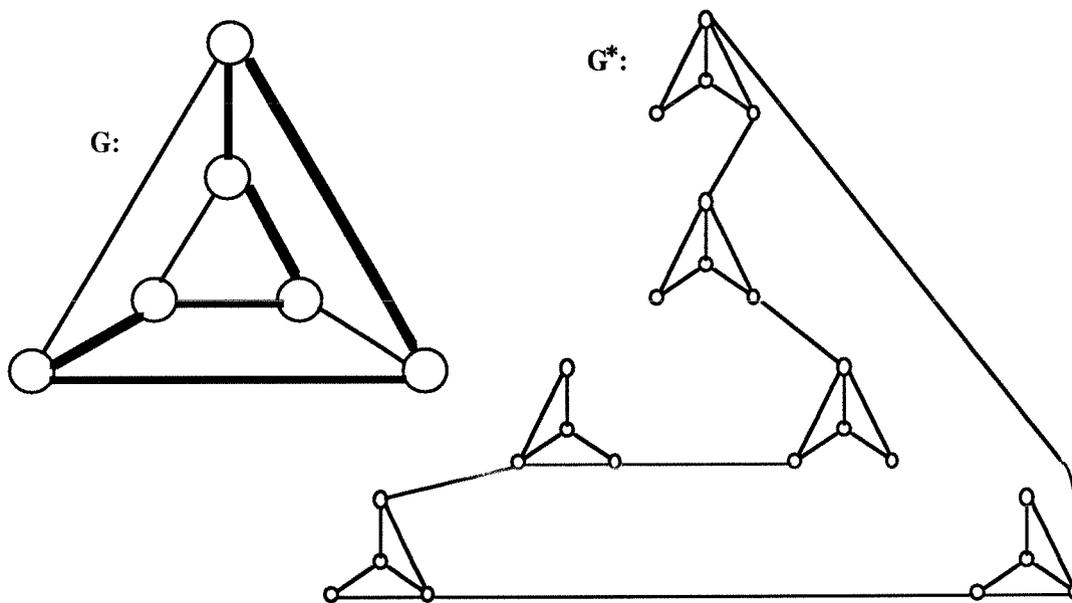


Figure III.B.1b

(and dismissed) using an algebraic argument.

In the five edge case, we can employ a distinction based on whether or not the "missing" edge in the module is incident to the infinite face of the module.

Suppose an infinite face edge is missing. Then the structure that results appears in figure III.B.1c. But since there is only one edge disjoint path from x to z going through the module, we see by hypothesis (biconnectedness) that a path exists from x to z independent of the stated module (denoted by the dashed, bold line). But then a K_4 homeomorph is evident on vertices a , b , c , and d , and so G^- cannot be series-parallel.

Now, suppose an edge incident to the central (not incident to the infinite face) vertex is missing, as shown in figure III.B.1d. Then by hypothesis there must exist two edge disjoint paths from x to y , and two from x to z . But then a K_4 homeomorph is evident on vertices a , b , c , and x , and we deny again that G^- is series-parallel.

So, each module has 2 or 3 A^- edges incident to it, and further, a module with 3 such incident edges can have no more than 4 B^- edges inside it.

Now we employ an algebraic argument to show that $|A^-|$ must be exactly $|V|$. Recall that $|B^-| \leq 5|V|$, so $|A^-| \geq |V|$ must hold in order that $|E^-| \geq 6|V|$ is satisfied.

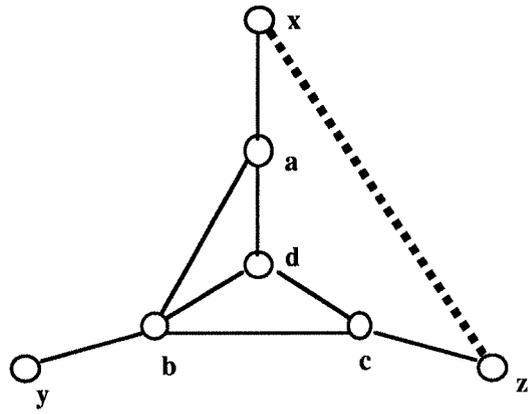


Figure III.B.1c

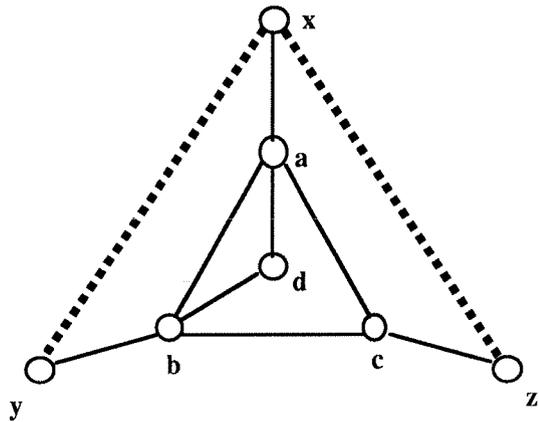


Figure III.B.1d

Let us establish the following notation:

$a \triangleq$ the number of A' edges in A^- exceeding $|V|$.

$c \triangleq$ the number of degree-2 modules in G^- .

$d \triangleq$ the number of degree-3 modules in G^- .

Now suppose $|A^-| = |V| + a$. Then there must be $2(|V| + a)$ edge-module incidences for the intermodular edges in A^- .

Clearly, the number of edge-module incidences for A^- edges is also equal to three times the number of degree-3 modules in G^- plus twice the number of degree-2 modules in G^- or

$$2|V| + 2a = 2c + 3d \quad (1)$$

Also, since every module must be either of degree-2 or degree-3, we must have

$$|V| = c + d \quad (2)$$

Now upon substitution of (2) in (1) we have $2a = d$ and so for each edge by which $|A^-|$ exceeds $|V|$, we must have two degree-3 modules in G^- in order to account for total degree in G^- .

Now recall that $|B^-| \leq 5|V|$ in general, but since we have established that degree-3 modules can have at most 4 B^- edges, we see that $|B^-| \leq 5|V| - d$ (reducing the bound by one for each degree-3 module in G^-) or equivalently,

$$|B^-| \leq 5|V| - 2a \quad (3).$$

We initially supposed that $|A^-| = |V| + a$ (4) and also recall that $|A^-| + |B^-| = |E^-|$. Now we add (3) and (4), obtaining $|E^-| \leq 6|V| - a$. Therefore to satisfy $|E^-| \geq 6|V|$, we must have $a = 0$, so $|A^-| = |V|$. Now since $2a = d$, we see that $d=0$ which implies that there exist only degree-2 modules in G^- . Since G^- is biconnected and each module in G^- is incident to two A^- edges, these edges correspond to a hamiltonian cycle in G . It is certainly easy to test if a graph is biconnected and series-parallel and so the proof is complete.

■

It should be noted that the above is an independent proof of a result given in [1].

The following corollary is immediate.

Corollary: P_{SPSUB} remains hard on planar graphs.

■

2. Halin

It would not be surprising that a result similar to the theorem in section III.B.1 also existed in the case of Halin subgraphs. The corresponding problem is:

P_{HS} : Given a graph $G=(V,E)$ and an integer j , does there exist a Halin subgraph of G , $G'=(V,E')$ where $E' \subseteq E$ and with $|E'| \geq j$?

Following, we show that resolving P_{HS} is just as difficult as P_{SPSUB} .

Theorem: P_{HS} is ~~NP~~-complete.

Proof: We will show a reduction from the ~~NP~~-complete problem of finding a longest cycle the statement of which appears below [7]:

P_C : Given a planar graph $G=(V,E)$ and an integer $|E| \geq k \geq 3$, does there exist a cycle in the graph of length at least k ?

From an instance of P_C let us create an instance of P_{HS} as indicated in Figure III.B.2a. Add a vertex to every edge in G and also add a "supervertex", v_x . Connect the supervertex to every vertex of the stated homeomorph of G . Set $j = 4k$.

Let the constructed graph be $G'=(V',E')$, where

$V^* \triangleq$ Vertices inserted on each edge in E

$v_x \triangleq$ The supervertex

$V' \triangleq V \cup V^* \cup v_x$

$E^* \triangleq$ Edges formed by the "division" of E edges by V^*

$E_x \triangleq$ Edges connecting v_x vertex to $V \cup V^*$ vertices

$E' \triangleq E^* \cup E_x.$

Note that:

$$|V^*| = |E|$$

$$|V'| = |V| + |E| + 1$$

$$|E^*| = 2|E|$$

$$|E_x| = |V| + |E|$$

$$|E'| = 3|E| + |V|.$$

Now assume there exists a cycle in G of length at least k . Clearly, $k \leq |V|$. By construction, a cycle of length k in G implies a cycle of length $2k$ in G' , since each E edge was split to form two E^* edges and one V^* vertex. Now this cycle in G' passes through $2k$ vertices in G' , each of which is connected to v_x by an edge in E_x . These $2k$ edges form a star graph with $2k + 1$ vertices with v_x at the "hub". Adding the $2k$ cycle edges produces a wheel graph and hence the desired Halin subgraph of size $4k$. This construction is demonstrated in Figure III.B.2b.

Now suppose there exists a halin subgraph of G' , say $G' = (V', E')$ where $V' \subseteq V'$ and $E' \subseteq E'$ and with edge cardinality at least $4k$. Observe that each element of E' must be either a **cycle** edge in E^* , a **tree** edge in E^* , or **out** of E^* ($\in E' \setminus E^*$).

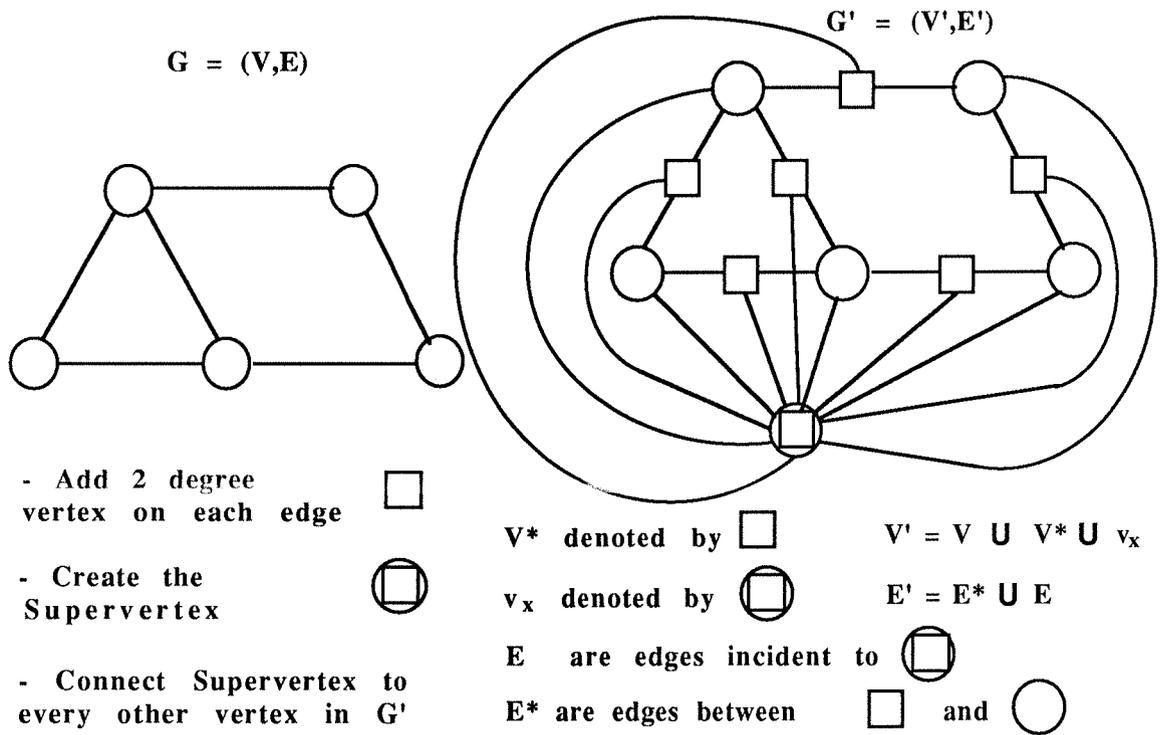


Figure III.B.2a

Example with cycle size = 4
 Cycle indicated in bold print

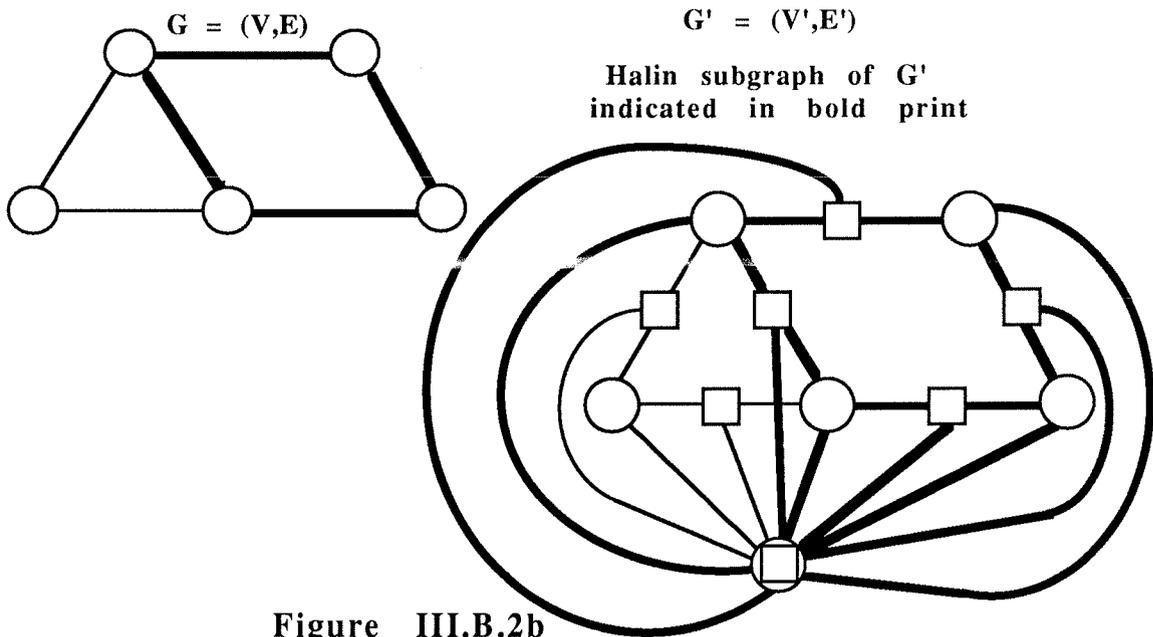


Figure III.B.2b

First we establish that v_x must be in V^- . Suppose otherwise; i.e. that $v_x \notin V^-$. Then none of the E_x edges can be in the stated subgraph. This leaves each V^* vertex in G' incident to only two edges. But no vertex in a halin graph can be of degree less than three, so these vertices and the edges incident to them, E^* , also cannot be in the stated subgraph. This leaves no edges at all, so no halin graph is possible, contrary to hypothesis. Therefore, $v_x \in V^-$.

Now we establish that v_x is not a pendant. Suppose otherwise. Then the hypothesized **cycle** passes through v_x , implying that exactly two of the edges in E_x are **cycle** edges, and exactly one edge in E_x is a **tree** edge. Now, except for trivial cases, the **cycle** must pass through at least one V^* vertex that is not connected directly to v_x by a **cycle** edge. Then, the remaining edge incident to that V^* vertex must be a **tree** edge in E^- . But that edge connects the stated V^* vertex directly to v_x which denies that G^- is Halin (see Figure III.B.2c). Therefore, v_x cannot be a pendant, which means it has no **cycle** edges incident to it.

Now in G' , v_x is connected to every other vertex by an edge in E_x . Since $v_x \in V^-$, one of these E_x edges is a **tree** edge connecting v_x to another vertex y in G' . Then there are four possibilities: (a) y is in V and is a pendant, (b) y is in V and is not a pendant, (c) y is in V^* and is a pendant, or (d) y is in V^* and is not a pendant. We shall

examine cases (b) and (d) now and show that they are impossible.

- case (b). Since we suppose that y is not a pendant, at least two E^* edges incident to y must also be **tree** edges. Both of these edges lead to degree-3 V^* vertices. Pick one of these vertices, say p . If p is a pendant, then both of the other two edges incident to it must be **cycle** edges. But one of these edges leads back to v_x which contradicts that v_x is not a pendant. If p is not a pendant, then both of the other two edges incident to it are **tree** edges, but the one that leads back to v_x forming a cycle among the edges picked thus far and G could not be Halin. Therefore, case (b) is impossible.

- case (d). Since we suppose that y is not a pendant and y is in V^* , both of the other two edges incident to y must also be **tree** edges. These two edges lead to V vertices a and b . Now suppose a and b are both pendants. Then the **cycle** passes through both a and b . This defines two edge-disjoint simple paths from a to b that do not pass through non-pendant vertices v_x and y . Furthermore, since a and b are both in V , there must be at least one vertex in V^* on each of these paths. Name any V^* vertex on one path u , and any V^* vertex on the

other path v . Now, since u is of degree three in the subgraph, the non-**cycle** edge incident to u (which leads back to v_x) must be a **tree** edge. The same is true for vertex v . However, this induces a subgraph homeomorphic to $K_{3,3}$, since the six specified vertices can be partitioned into two sets, as $\{v_x, a, b\}$ and $\{y, u, v\}$ accordingly (see Figure III.B.2d). Similarly, if one or both of a and b are not pendants, then the additional **tree** edges required to connect them to pendants will again produce a $K_{3,3}$ homeomorph when added to the previous construction. In either case we deny planarity in the hypothesized subgraph. Therefore, case (d) is impossible.

Now we have contradicted each case except those where vertex y is a pendant. Since y was chosen arbitrarily, we conclude that all vertices in the Halin subgraph except vertex v_x must be pendant vertices. This can only occur with the **tree** edges forming a star with the "hub" v_x . Clearly, such a (Halin) graph has exactly half of its edges in the **cycle** and half in the **tree**, all of the latter incident to the "hub" vertex v_x . Since we have supposed the existence of a halin subgraph with $|E^-| \geq 4k$, we see that at least $2k$ edges must form the **cycle** in G^- . This cycle is represented by a vertex sequence alternating between

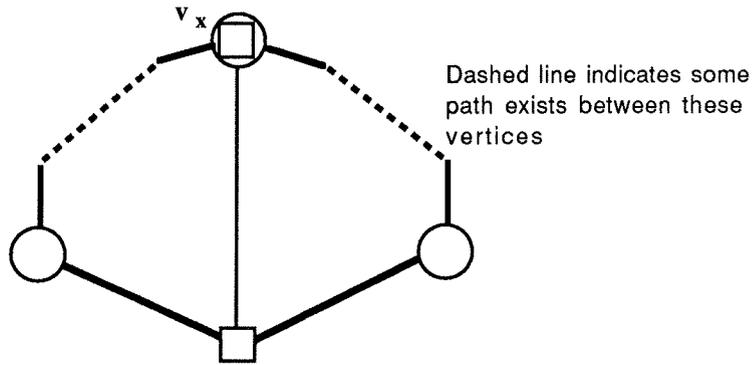


Figure III.B.2c

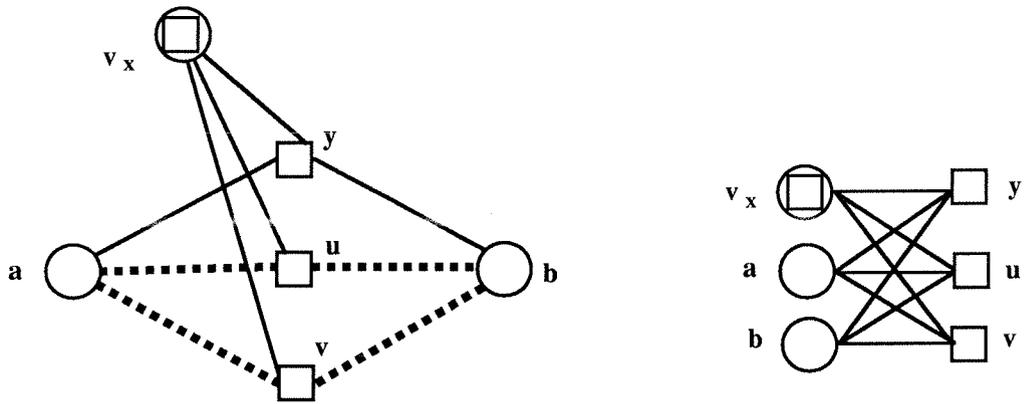


Figure III.B.2d

vertices in V and those in V^* . But recall that during the construction of G' , each edge of G was "split" into two edges by the insertion of a V^* vertex. Thus, the **cycle** portion of G' having at least $2k$ edges corresponds exactly to a cycle in G of length at least k .

We can test for the Halin property in polynomial time and the result of the theorem follows.

■

An easy corollary results by fixing k in the theorem above. Letting $k = |V|$ we have:

Corollary: Deciding if $G=(V,E)$ possesses a **spanning** Halin subgraph is **NP**-complete.

■

Realistically, our two subgraph results are somewhat expected following a pattern exhibited for similar problems. On the other hand, the analogous questions regarding supergraphs might be more interesting. In the next section, we show this to be the case.

C. Supergraph Results

As discussed earlier, the supergraph version is not meaningful for series-parallel graphs. Thus we will consider only the following problem:

P_H : Given a graph $G=(V,E)$, does there exist a set of edges $E^+ \supseteq E$, such that the supergraph $G^+=(V,E^+)$ is Halin?

Recall that we may assume that G is not Halin since testing for this property is easy. We may also assume that G is not 3-connected, following from the fact that Halin graphs are 3-connected and minimal in this regard.

Our next result suggests that resolving P_H is no easier than the Halin subgraph version.

Theorem: P_H is ~~NP~~-complete.

Proof: We will show a reduction from the (strong sense) ~~NP~~-complete problem of 3-Partition the statement of which appears below [7].

P_{3P} : Given a set A of $3m$ elements, an integer bound B , and an integer size $s(a)$ for each $a \in A$ such that $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$, can A be partitioned into m disjoint sets A_1, A_2, \dots, A_m such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$?

(Observe that each A_i must contain exactly three elements from A).

From an instance of P_{3P} let us create an instance of P_H as indicated in figure III.C.1. Attached to vertex t are

"tails" which correspond to the elements in A with the length of each tail related to the size of the respective element. The upper part of the graph and in particular the m "segments" correspond to the sets A_1, A_2, \dots, A_m , each with size B where the latter is denoted by the B vertices inserted within each segment as indicated. We shall use the terms "intersegment" vertex and "intra-segment" vertex to address these vertices near the top of the figure, as shown. Certainly, G in figure III.C.1 is not Halin nor is it 3-connected.

Now, given a suitable 3-partition of A, we can construct a "Halin completion" of G in the following manner. For each A_i , place the three relevant tails in an interior face bounded by intersegment vertices, intra-segment vertices, and t (we call each of these faces a "sector"). From each pendant (degree-1) vertex of the tails create two edges from the pendant to an adjacent pair of intra-segment vertices. For each (if any) other (degree-2) vertex on a tail, create one edge from the tail to an intra-segment vertex. In this way, we add $\sum_{a \in A_i} (s(a)-1) + 3 = B$ edges.

It is easy to see that planarity can be maintained in this connection. The construction is thus complete yielding a Halin graph $H=(V, E_H)$ with E_H defined by E augmented with the new edges described; the **cycle** edges are those defining

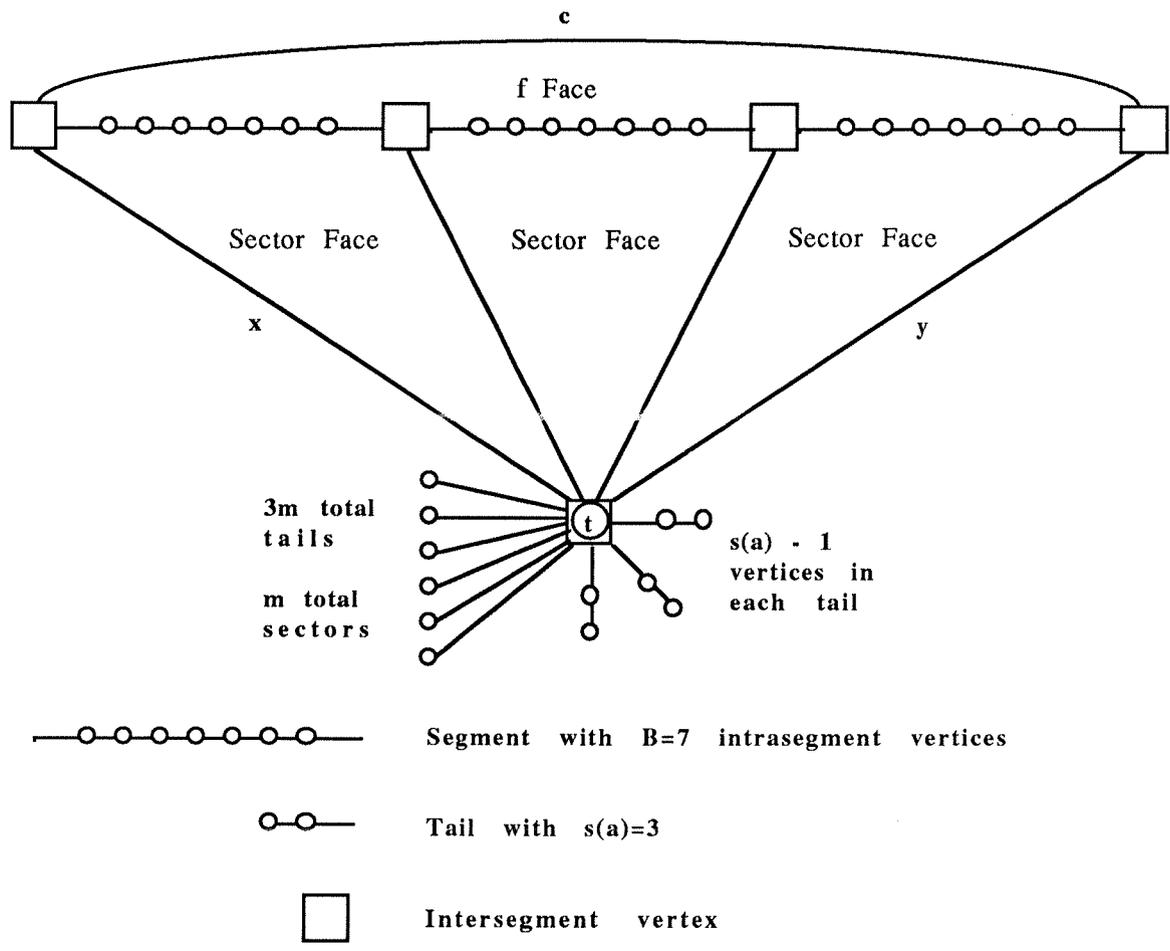


Figure III.C.1

face f , say E_f , and the **tree** edges are given by $E_H \setminus E_f$.

Figure III.C.2 demonstrates this.

Conversely, assume there exists a Halin completion of G , say $G^c = (V, E^c)$. Let $E^c = E \cup E'$. Vertex t must be a non-pendant vertex, since its degree in G exceeds 3 (we avoid trivialities in the statement of P_{3p}). Thus, edges x and y are **tree** edges implying the edge c is a **cycle** edge.

Accordingly, the intersegment vertices incident to edge c must be pendants, which implies that edges a and b are **cycle** edges as well. But this means that the **cycle** component in G^c is either defined by face f of G or a subface of f created in G^c . Accordingly, the tails of G must be part of the **tree** in G^c . Observe also that the upper portion of G exscribed by edges x , y , and c is 3-connected and we therefore may assume an embedding of it as shown in figures III.C.1 and III.C.2.

If the number of **cycle** edges in $G^c = k$, we must have that $k \leq \dim(f) \leq m(B+1) + 1$, where $\dim(f)$ denotes the dimension of face f . Also recall that k has a natural lower bound:

$$k \geq |V|/2 + 1 = (2(m(B-1) + 1))/2 + 1 = m(B-1) + 2.$$

But the vertices in the tails of G have degree less than 3 and therefore have total deficiency at least mB . This

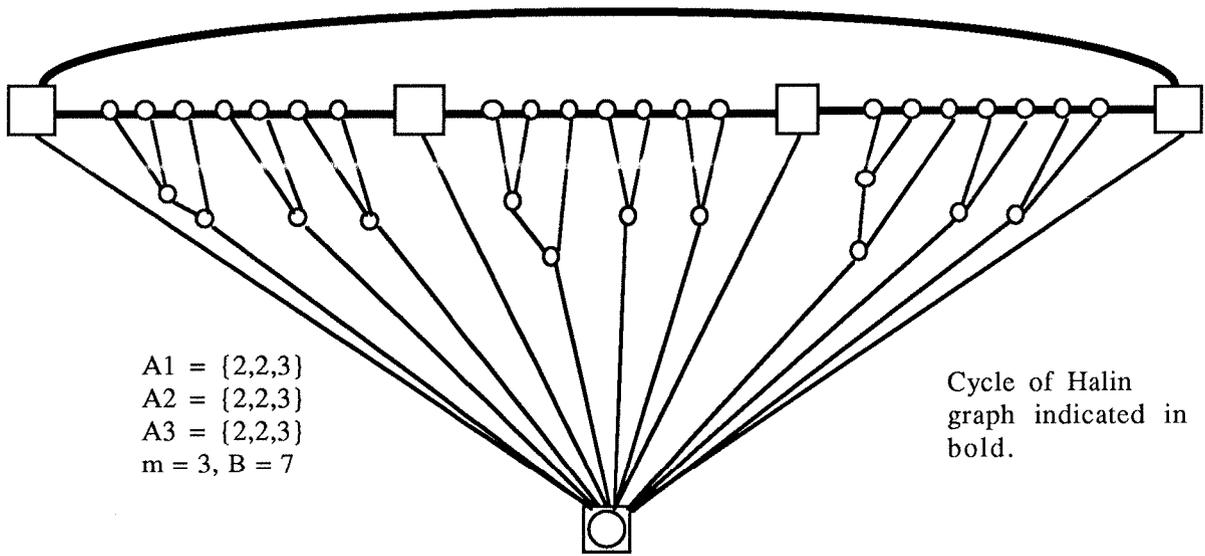


Figure III.C.2

deficiency has to have been satisfied by edges attached to the tail vertices but not extending between distinct tails nor between distinct vertices of the same tail which in either case denies that the tails are part of a tree.

Hence, $|E'| \geq mB$.

On the other hand, the intrasegment vertices in G all have degree 2 and thus a deficiency of mB as well. But in G we have $|E| = 2mB - m + 2$ and in any Halin graph of order p with a cycle of length k , we have $p + k - 1$ edges. In our construction, G^c must then have size $2(m(B-1) + 1) + k - 1$. Letting θ be the number of edges to add to G to create G^c , it is easy to see that $\theta = k - m - 1$. Let us suppose that k is different from its upper bound of $m(B+1) + 1$. Then $\theta < mB$ and E' cannot be formed as required. Hence, $k = m(B+1) + 1$ and the cycle in G^c is defined explicitly by face f . Thus all the vertices on f are pendant vertices and their deficiency is exactly mB . The only way this deficiency can be satisfied is by edges connected to tail vertices. Exactly B of these are required in each sector face and these edges connect exactly those vertices of the tails which must be embedded in the sectors. Moreover, if any tail vertex is connected to a vertex in a given sector so must every other vertex in that tail since G^c is planar. Thus, every one of the m sectors has exactly three tails from G embedded within it with each connected by exactly B

edges to the respective vertices on face f . But then each of these sector-tail embeddings forms a triple which corresponds to a suitable 3-partition of A .

The transformation from P_{3p} to form G is valid following the strong sense status of P_{3p} . This along with P_H 's inclusion in \mathbf{NP} yields the desired result.

■

CHAPTER IV

CONCLUSIONS

This thesis has addressed some of the issues associated with graph approximation. In Chapter II, we demonstrated some anomalous behavior of possible approximation schemes. This suggested to us that finding appropriate approximating graphs might be difficult. The results of Chapter III show this to be the case. Halin graphs, in particular, were shown to pose significant problems when used for approximation. This was somewhat surprisingly true even for the Halin supergraph case, despite the fact that finding series-parallel supergraphs is generally less interesting. In conclusion, although we began this effort seeking approximation strategies, our research led us in the direction of the negative but strong results of Chapter III.

There is room for further research into graph approximation. Certainly other recursive graph classes could be explored in the manner that we investigated Halin and series-parallel graphs. Alternatively, liberalized definitions of subgraph and supergraph could be used, allowing vertices to be removed and added along with edges. Finally, the notions of subgraph and supergraph could be

combined, allowing some edges and/or vertices to be removed and others to be added in the same approximation.

APPENDIX

A. Complete Solution Methodology for P_{SES}

Here we present the multiplication tables used to solve P_{SES} on Halin graphs. The following guidelines will assist the reader in using the tables: Read G_1 on left column, and G_2 in first row. The "offspring" graph that results from those two parents in that order is indicated in the table. A "-" indicates this composition not possible.

Operation Type 1, $\{(3,3,3)\}$ (see Figure APP.A.1)

Table APP.A.1 Composition Rules for Operation Type 1

$G_2 \rightarrow$ $G_1 \downarrow$	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK
SES	SES	LTBI	LTBI	SES	SES	-	LFRK	-	LFRK	-	-
TRI	RTBI	-	TRI	-	TRI	-	-	-	RTPA	-	-
LTBI	SES	-	LTBI	-	LTBI	-	-	-	LFRK	-	-
RTBI	RTBI	TRI	TRI	RTBI	RTBI	-	RTPA	-	RTPA	-	-
LRBI	SES	TRI	LTBI	RTBI	LRBI	-	RTPA	-	LFRK	-	-
LTPA	RFRK	-	LTPA	-	LTPA	-	-	-	TFRK	-	-
RTPA	-	-	-	-	-	TRI	-	RTPA	-	RTPA	RTBI
LRPA	-	-	-	-	-	LTPA	-	LRPA	-	TFRK	RFRK
LFRK	-	-	-	-	-	LTBI	-	LFRK	-	LFRK	SES
TFRK	-	-	-	-	-	LTPA	-	TFRK	-	TFRK	RFRK
RFRK	RFRK	LTPA	LTPA	RFRK	RFRK	-	TFRK	-	TFRK	-	-

Operation Type 2, $\{(3,3,4)\}$ (adding K_2 to the TOP terminal, RIGHT terminal of K_2 (G_1) connects to TOP terminal of G_2 - see Figure APP.A.2).

Table APP.A.2 Composition Rules for Operation Type 2

$G_2 \rightarrow$ $G_1 \downarrow$	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK
IN	-	-	-	-	-	LTPA	RTPA	-	LFRK	-	RFRK
OUT	LRBI	-	TRI	TRI	-	-	-	-	-	LRPA	-

Operation Type 3, $\{(3,3,4)\}$ (adding K_2 to the LEFT terminal, RIGHT terminal of K_2 (G_1) connects to LEFT terminal of G_2 - see Figure APP.A.3).

Table APP.A.3 Composition Rules for Operation Type 3

$G_2 \rightarrow$ $G_1 \downarrow$	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK
IN	-	-	-	-	-	LTPA	-	LRPA	-	TFRK	RFRK
OUT	RTBI	-	TRI	-	TRI	-	-	-	RTPA	-	-

Operation Type 4, $\{(3,3,4)\}$ (adding K_2 to the RIGHT terminal, LEFT terminal of K_2 (G_1) connects to RIGHT terminal of G_2 - see Figure APP.A.4). Note that in this case, G_2 is on the left and G_1 is on the right.

Table APP.A.4 Composition Rules for Operation Type 4

$G_2 \rightarrow$ $G_1 \downarrow$	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK
IN	-	-	-	-	-	-	RTPA	LRPA	LFRK	TFRK	-
OUT	LTBI	-	-	TRI	TRI	-	-	-	-	-	LTPA

Operation Type 5, $\{(3,3,3)\}$ (adding K_2 between TOP and LEFT terminals of G_2 - see Figure APP.A.5).

Table APP.A.5 Composition Rules for Operation Type 5

$G_2 \rightarrow$ $G_1 \downarrow$	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK
IN	RFRK	LTPA	LTPA	RFRK	RFRK	LTBI	TFRK	LFRK	TFRK	LFRK	SES
OUT	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK

Operation Type 6, $\{(3,3,3)\}$ (adding K_2 between TOP and RIGHT terminals of G_2 - see Figure APP.A.6).

Table APP.A.6 Composition Rules for Operation Type 6

$G_2 \rightarrow$ $G_1 \downarrow$	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK
IN	LFRK	RTPA	LFRK	RTPA	LFRK	TFRK	RTBI	RFRK	SES	RFRK	TFRK
OUT	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK

Operation Type 7, $\{(3,3,3)\}$ (adding K_2 between LEFT and RIGHT terminals of G_2 - see Figure APP.A.7).

Table APP.A.7 Composition Rules for Operation Type 7

$G_2 \rightarrow$ $G_1 \downarrow$	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK
IN	TFRK	LRPA	TFRK	TFRK	LRPA	LFRK	RFRK	LRBI	RFRK	SES	LFRK
OUT	SES	TRI	LTBI	RTBI	LRBI	LTPA	RTPA	LRPA	LFRK	TFRK	RFRK

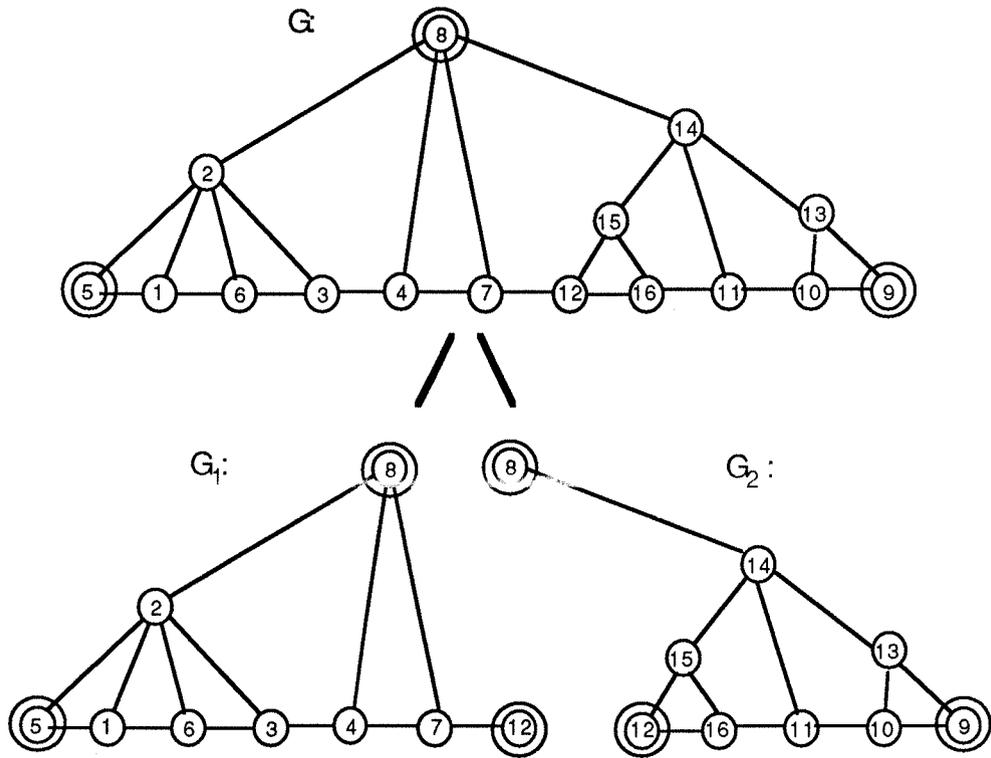


Figure APP.A.1

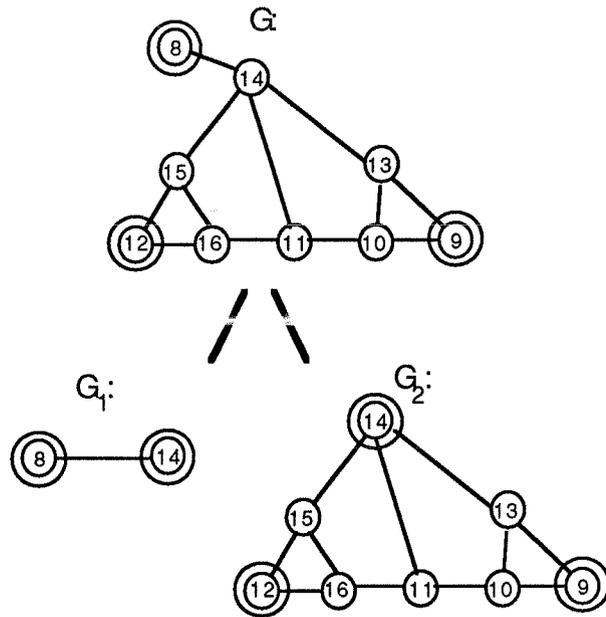


Figure APP.A.2

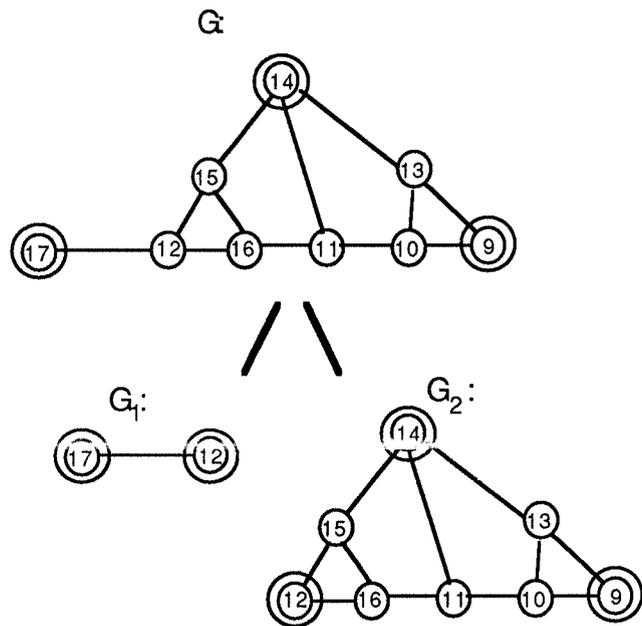


Figure APP.A.3

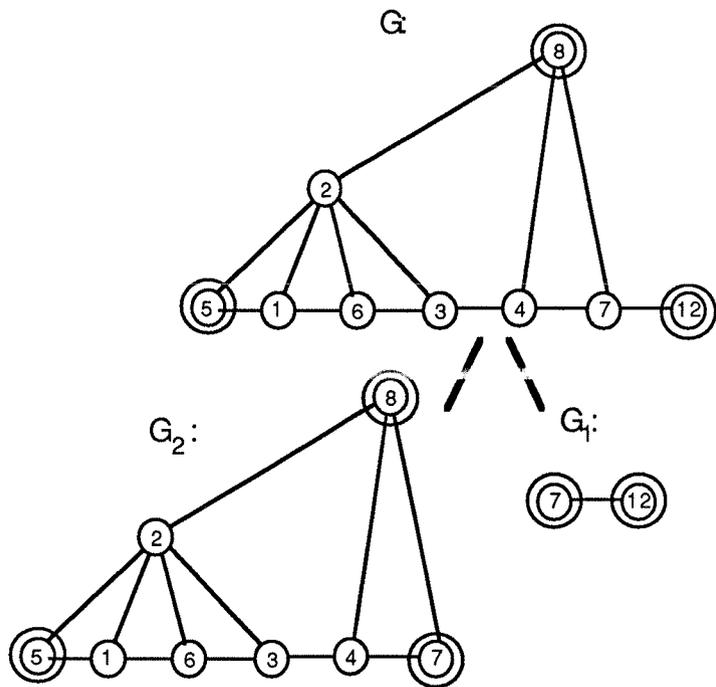


Figure APP.A.4

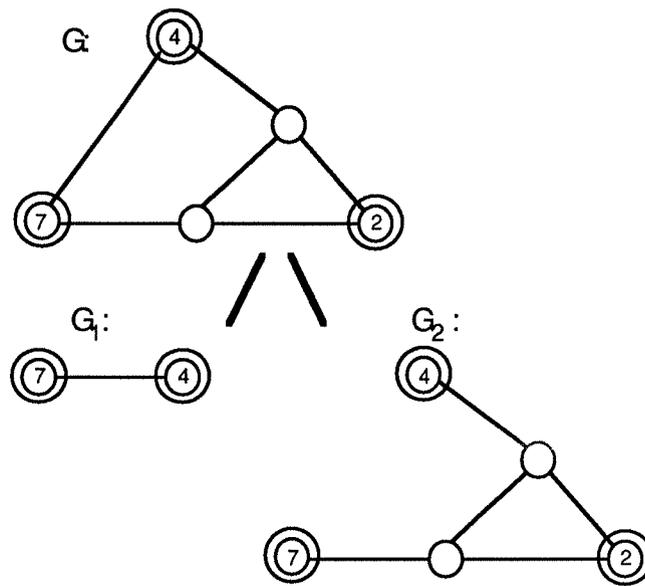


Figure APP.A.5

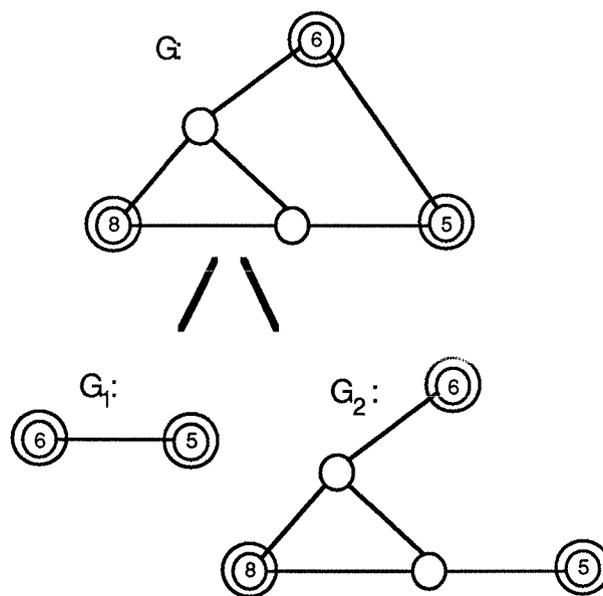


Figure APP.A.6

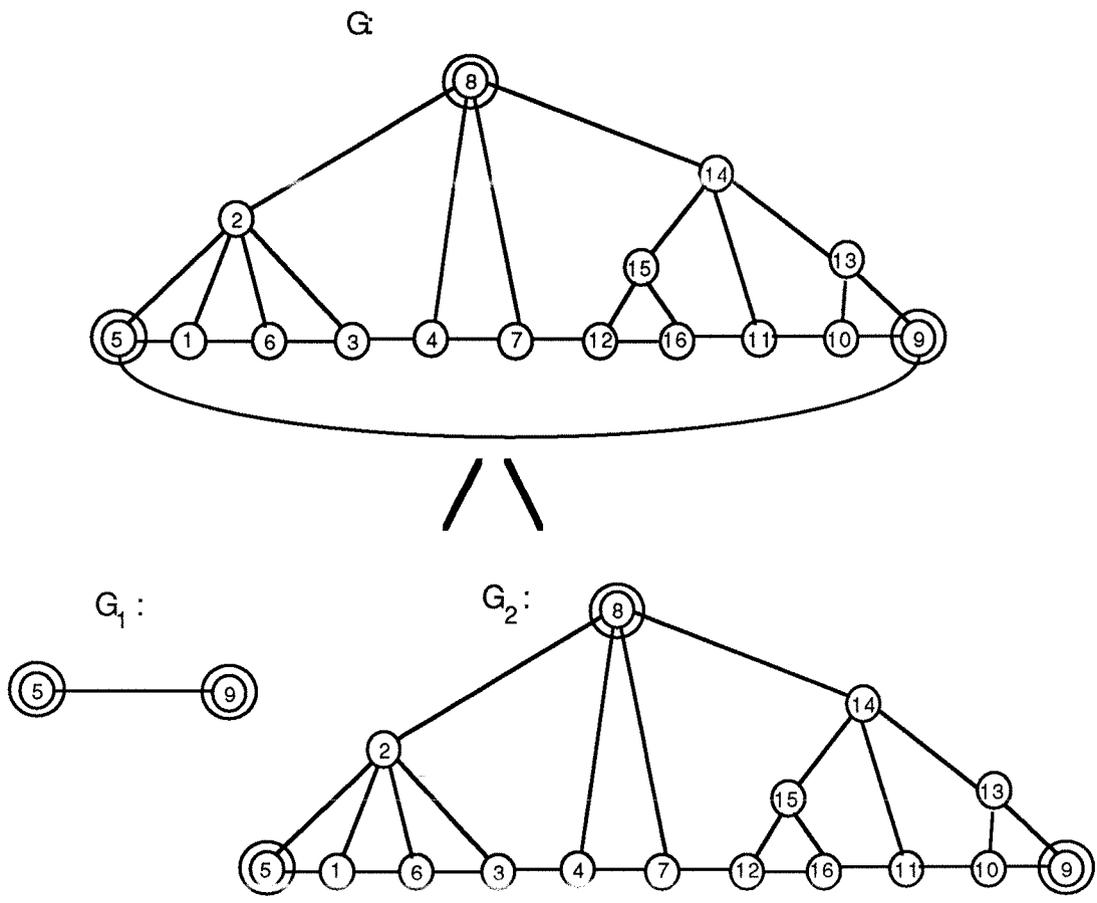


Figure APP.A.7

B. Complete Solution Methodology for P_{vc}

In this section we present solution methodology for P_{vc} on Halin graphs, and we show the solution to an example problem.

First we observe that we use the same "operations" for P_{vc} as we did for P_{SES} . Now since every vertex in a vertex cover is simply either "in" the cover or "out" of the cover, we see that our data storage system only needs to keep track of the status of each terminal vertex with respect to the above property. We also note that the only initial elements for our construction will be K_2 and K_3 (K_3 again is used only as a computational convenience, since K_3 easily reduces to edges). Thus we establish **No, L, R, and B** to indicate no vertex, the left vertex, the right vertex, and both vertices, respectively, as being "in" the cover for the K_2 case. Similarly, we let **No, L, R, T, LR, LT, RT, and LRT** denote all eight possible permutations of the three terminal case with respect to the above property.

Next we establish initialization rules for our two initial cases. Since we are attempting to find the minimum cardinality vertex cover, we observe that to determine the "impact" of a subgraph on our objective function, we simply need to count vertices "in" the cover, and each vertex so counted adds 1 to the value of the objective function, while

an "out" vertex adds 0. Where a vertex cover is not possible, the value ∞ is assigned. Thus for the K_2 case:

$$\begin{aligned} P_{No}[K_2] &= \infty \\ P_L[K_2] &= 1 \\ P_R[K_2] &= 1 \\ P_B[K_2] &= 2 \end{aligned}$$

For the K_3 case, we have:

$$\begin{aligned} P_{No}[K_3] &= \infty \\ P_L[K_3] &= \infty \\ P_R[K_3] &= \infty \\ P_T[K_3] &= \infty \\ P_{LR}[K_3] &= 2 \\ P_{LT}[K_3] &= 2 \\ P_{RT}[K_3] &= 2 \\ P_{LRT}[K_3] &= 3 \end{aligned}$$

Note that by substituting vertex weights for the above values, this method could be used to solve the minimum weighted vertex cover problem on halin graphs.

Now we consider the seven "Operation Types" and develop tables to determine what "offspring" graph results from each possible "parent" configuration.

Operation Type 1, $\{(3,3,3)\}$ (see Figure APP.A.1).

Table APP.B.1 Composition Rules for Operation Type 1

$G_2 \rightarrow G_1$	No	L	R	T	LR	LT	RT	LRT
No	No	-	R	-	-	-	-	-
L	L	-	LR	-	-	-	-	-
R	-	No	-	-	R	-	-	-
T	-	-	-	T	-	-	RT	-
LR	-	L	-	-	LR	-	-	-
LT	-	-	-	LT	-	-	LRT	-
RT	-	-	-	-	-	T	-	RT
LRT	-	-	-	-	-	LT	-	LRT

Operation Type 2, $\{(3,3,4)\}$ (adding K_2 to the TOP terminal, RIGHT terminal of K_2 (G_1) connects to TOP terminal of G_2 , see Figure APP.A.2).

Table APP.B.2 Composition Rules for Operation Type 2

$G_2 \rightarrow$ $G_1 \downarrow$	No	L	R	T	LR	LT	RT	LRT
No	No	L	R	-	LR	-	-	-
L	T	LT	RT	-	LRT	-	-	-
R	-	-	-	No	-	L	R	LR
B	-	-	-	T	-	LT	RT	LRT

Operation Type 3, $\{(3,3,4)\}$ (adding K_2 to the LEFT terminal, RIGHT terminal of K_2 (G_1) connects to LEFT terminal of G_2 , see Figure APP.A.3).

Table APP.B.3 Composition Rules for Operation Type 3

$G_2 \rightarrow$ $G_1 \downarrow$	No	L	R	T	LR	LT	RT	LRT
No	No	-	R	T	-	-	RT	-
L	L	-	LR	LT	-	-	LRT	-
R	-	No	-	-	R	T	-	RT
B	-	L	-	-	LR	LT	-	LRT

Operation Type 4, $\{(3,3,4)\}$ (adding K_2 to the RIGHT terminal, LEFT terminal of K_2 (G_1) connects to RIGHT terminal of G_2 , see Figure APP.A.4). Note: In this case, G_2 is on the left and G_1 is on the right.

Table APP.B.4 Composition Rules for Operation Type 4

$G_2 \rightarrow$ $G_1 \downarrow$	No	L	R	T	LR	LT	RT	LRT
No	No	L	-	T	-	LT	-	-
L	-	-	No	-	L	-	T	LT
R	R	LR	-	RT	-	LRT	-	-
B	-	-	R	-	LR	-	RT	LRT

Operation Type 5, $\{(3,3,3)\}$ (adding K_2 between TOP and LEFT terminals of G_2 , see Figure APP.A.5).

Table APP.B.5 Composition Rules for Operation Type 5

$G_2 \rightarrow$ $G_1 \downarrow$	No	L	R	T	LR	LT	RT	LRT
No	No	-	R	-	-	-	-	-
L	-	L	-	-	LR	-	-	-
R	-	-	-	T	-	-	RT	-
B	-	-	-	-	-	LT	-	LRT

Operation Type 6, $\{(3,3,3)\}$ (adding K_2 between TOP and RIGHT terminals of G_2 , see Figure APP.A.6).

Table APP.B.6 Composition Rules for Operation Type 6

$G_2 \rightarrow$ $G_1 \downarrow$	No	L	R	T	LR	LT	RT	LRT
No	No	L	-	-	-	-	-	-
L	-	-	-	T	-	LT	-	-
R	-	-	R	-	LR	-	-	-
B	-	-	-	-	-	-	RT	LRT

Operation Type 7, $\{(3,3,3)\}$ (adding K_2 between LEFT and RIGHT terminals of G_2 , see Figure APP.A.7).

Table APP.B.7 Composition Rules for Operation Type 7

$G_2 \rightarrow$ $G_1 \downarrow$	No	L	R	T	LR	LT	RT	LRT
No	No	-	-	T	-	-	-	-
L	-	L	-	-	-	LT	-	-
R	-	-	R	-	-	-	RT	-
B	-	-	-	-	LR	-	-	LRT

Table APP.B.8 Solution Summary to P_{vc} on Halin graph in Figure II.C.4. The decomposition tree for this graph is given in Figure APP.B.1.

Graph	Graph Properties
$P[G_1]$	$(\infty, \underline{1}, \underline{1}, 2)$
$P[G_2]$	$(\infty, \infty, \infty, \infty, 2, \underline{2}, 2, 3)$
$P[G_3]$	$(\infty, 2, \infty, 2, 3, 3, 3, \underline{3})$
$P[G_4]$	$(\infty, \underline{1}, \underline{1}, \underline{2})$
$P[G_5]$	$(\infty, \infty, \infty, 2, 3, 3, 3, \underline{3})$
$P[G_6]$	$(\infty, \underline{1}, \underline{1}, 2)$
$P[G_7]$	$(\infty, 3, \infty, 3, 4, \underline{3}, 3, 4)$
$P[G_8]$	$(\infty, \underline{1}, \underline{1}, 2)$
$P[G_9]$	$(\infty, \infty, \infty, 3, 4, \underline{3}, 3, 4)$
$P[G_{10}]$	$(\infty, \underline{1}, \underline{1}, \underline{2})$
$P[G_{11}]$	$(3, 3, 3, 4, 4, \underline{4}, 4, 5)$
$P[G_{12}]$	$(\infty, \underline{1}, \underline{1}, 2)$
$P[G_{13}]$	$(\infty, \infty, \infty, \infty, 2, \underline{2}, 2, 3)$
$P[G_{14}]$	$(\infty, \infty, 2, \underline{2}, 3, 3, 3, 3)$
$P[G_{15}]$	$(\infty, \infty, 5, 5, 5, \underline{5}, 5, 6)$
$P[G_{16}]$	$(\infty, \underline{1}, \underline{1}, 2)$
$P[G_{17}]$	$(5, 5, 6, 5, 6, 6, 6, \underline{6})$
$P[G_{18}]$	$(\infty, \underline{1}, \underline{1}, 2)$

Table APP.B.9 Solution Summary Continued

Graph	Graph Properties
P[G ₁₉]	($\infty, \infty, \infty, \infty, \underline{2}, 2, 2, 3$)
P[G ₂₀]	($\infty, 2, 2, \infty, 3, 3, 3, \underline{3}$)
P[G ₂₁]	($\infty, \underline{1}, 1, 2$)
P[G ₂₂]	($2, 3, 3, 3, 3, \underline{3}, 4, 4$)
P[G ₂₃]	($\infty, \underline{1}, 1, 2$)
P[G ₂₄]	($\infty, \infty, 3, 3, 3, \underline{3}, 4, 4$)
P[G ₂₅]	($\infty, 1, \underline{1}, 2$)
P[G ₂₆]	($3, 3, 4, 4, 4, 4, 4, \underline{4}$)
P[G ₂₇]	($\infty, \underline{1}, 1, 2$)
P[G ₂₈]	($\infty, \infty, \infty, \infty, \underline{2}, 2, 2, 3$)
P[G ₂₉]	($\infty, 2, 2, \infty, 3, 3, 3, \underline{3}$)
P[G ₃₀]	($5, 5, 5, 5, 5, 5, 5, \underline{5}$)
P[G ₃₁]	($\infty, 1, 1, \underline{2}$)
P[G ₃₂]	($5, 5, 5, 6, 5, 6, 6, \underline{6}$)
P[G ₃₃]	($10, 10, 10, 10, 10, 10, 10, \underline{10}$)
P[G ₃₄]	($\infty, 1, 1, \underline{2}$)
P[G ₃₅]	($\infty, 10, 10, \infty, 10, 10, 10, \underline{10}$)

As a result, we see that the minimum cardinality vertex cover for G has size 10, and one such cover consists of the following vertices: $\{2, 4, 5, 6, 8, 9, 10, 12, 14, 16\}$.

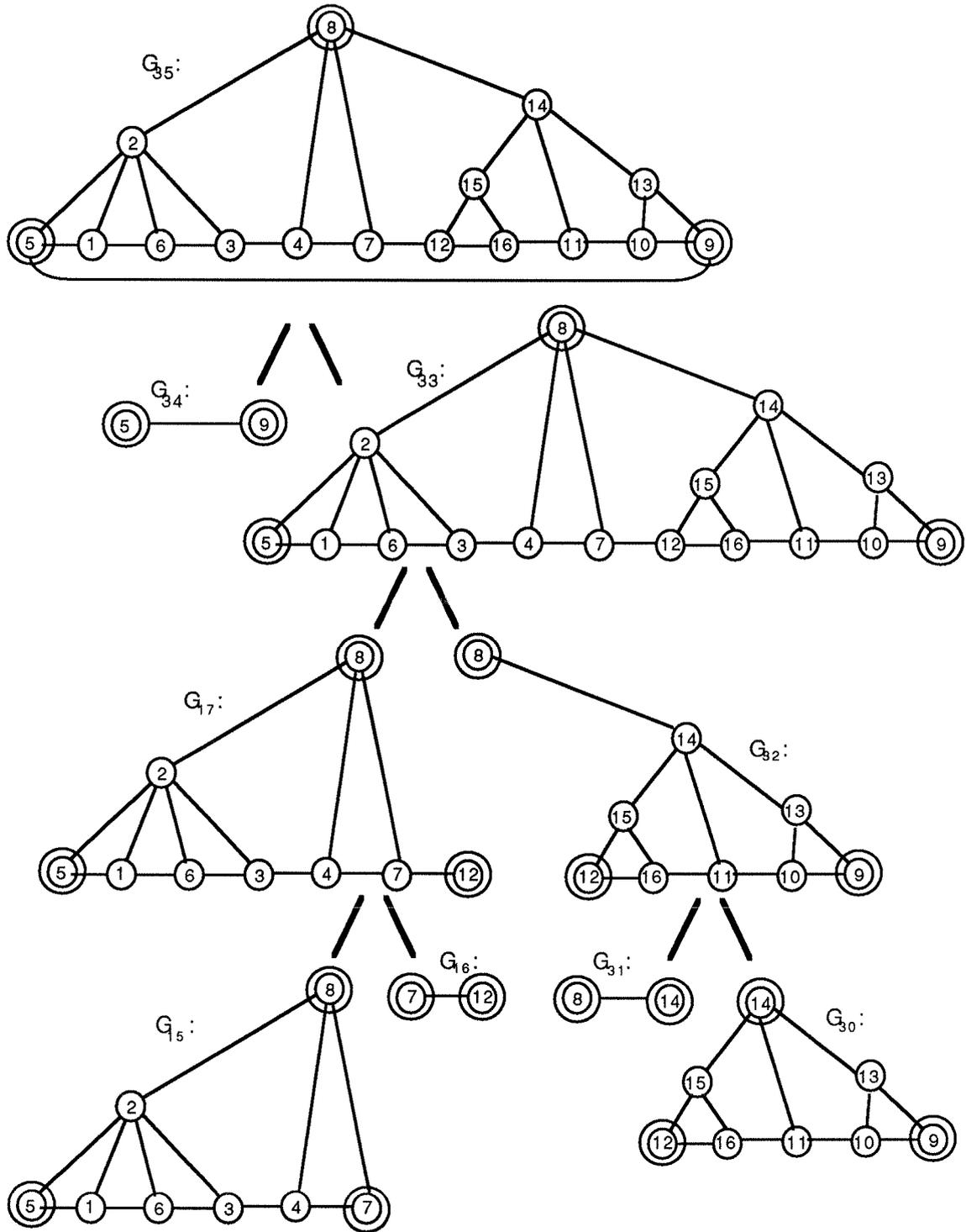


Figure APP.B.1

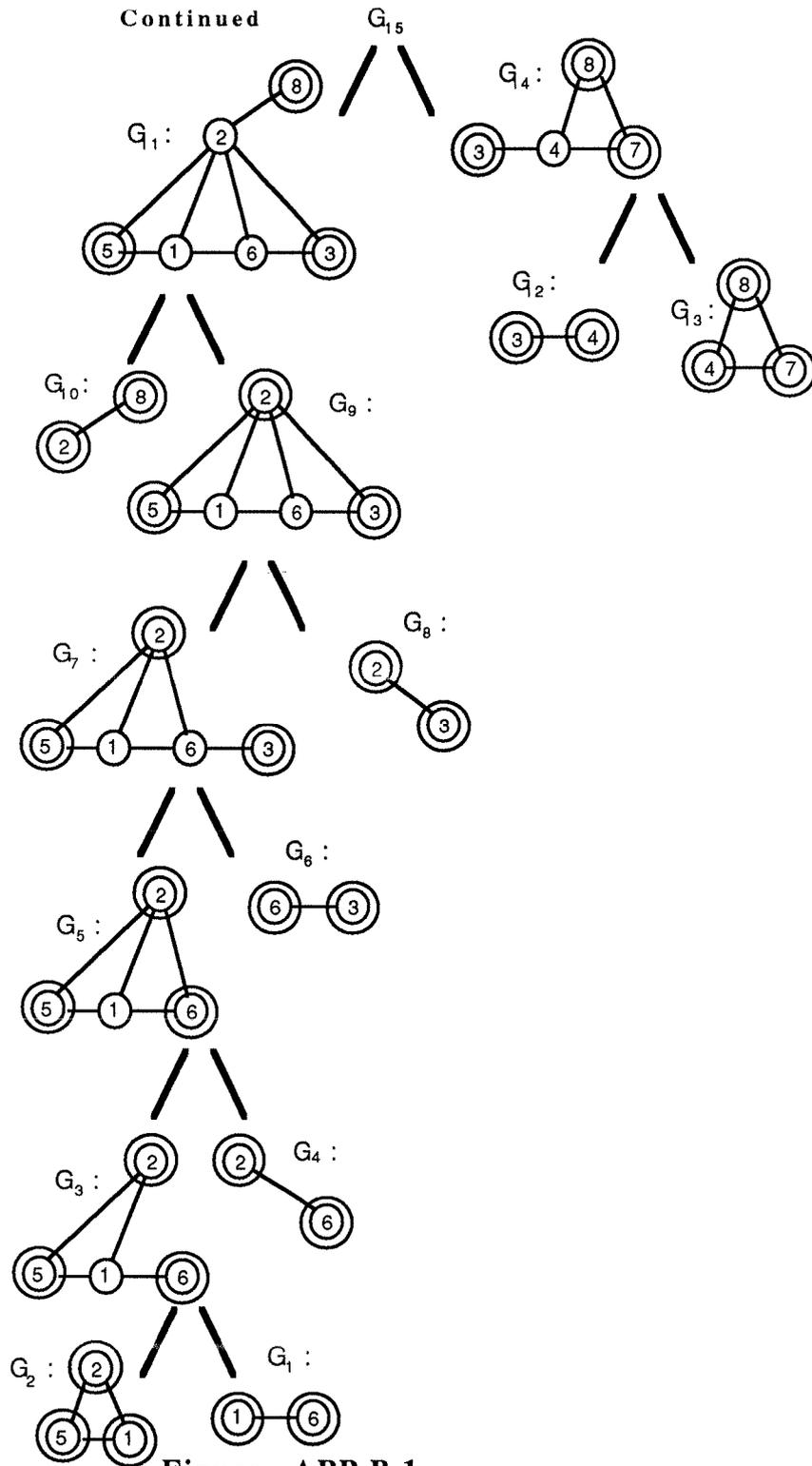


Figure APP.B.1

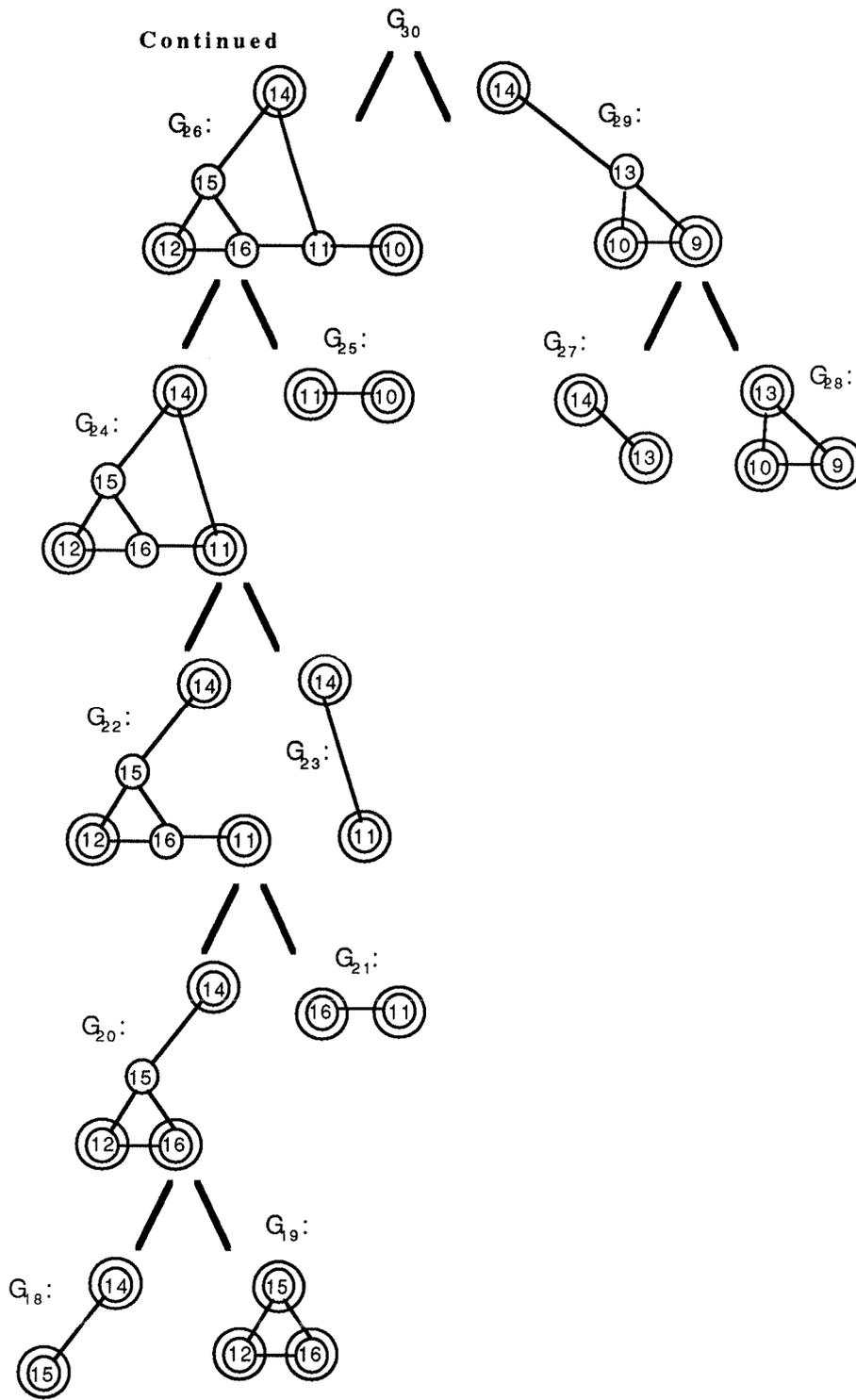


Figure APP.B.1

REFERENCES

- [1] Asano, T., "An Application of Duality to Edge-Deletion Problems," SIAM Journal of Computing, **16**, 312-331 (1987).
- [2] Borie, R. B., "Recursively Constructed Graph Families: Membership and Linear Algorithms," Ph.D. Thesis, School of Information and Computer Science, Georgia Institute of Technology, 1988.
- [3] Borie, R. B., R. G. Parker, C. A. Tovey, "Algorithms for Recognition of Regular Properties and Decomposition of Recursive Graph Families," Annals of Operations Research, to appear.
- [4] Borie, R. B., R. G. Parker, C. A. Tovey, "Deterministic Decomposition of Recursive Graph Classes", Discrete Mathematics, to appear.
- [5] Cornuéjols, G., D. Naddef, and W. R. Pulleybank, "Halin Graphs and the Travelling Salesman Problem," Mathematical Programming, **26**, 287-294 (1983).
- [6] Dirac, G. A., "A Property of 4-Chromatic Graphs and Some Remarks on Critical Graphs," Journal of the London Mathematical Society, **27**, 85-92 (1952).
- [7] Garey, M. R., D. S. Johnson, Computers and Intractability, W. H. Freeman and Company (1979).
- [8] Garey, M. R., D. S. Johnson, R. E. Tarjan, "The Planar Hamiltonian Circuit Problem is NP-Complete," SIAM Journal of Computing, **5**, 704-714 (1976).
- [9] Richey, M. B., "Combinatorial Optimization on Series-Parallel Graphs: Algorithms and Complexity," Ph.D. Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, 1985.
- [10] Richey, M. B. and R. G. Parker, "On Finding Spanning Eulerian Subgraphs," Naval Research Logistics Quarterly, **32**, 443-455 (1985).